DeTangle: A Framework for Interactive Prediction and Visualization of Gene Regulatory Networks

Doaa Abdelsalam Ahmed Mohamed Altarawy

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Lenwood S. Heath, Chair
Ruth Grene
Clifford A. Shaffer
Chris North
Mohamed Ismail

April 24, 2017
Blacksburg, Virginia

Keywords: Gene regulation, prior knowledge, gene regulatory network inference, visualization, machine learning.

DeTangle: A Framework for Interactive Prediction and Visualization of Gene Regulatory Networks

Doaa Altarawy

(ABSTRACT)

With the abundance of biological data, computational prediction of gene regulatory networks (GRNs) from gene expression data has become more feasible. Although incorporating other prior knowledge (PK), along with gene expression, greatly improves prediction accuracy, the accuracy remains low. PK in GRN inference can be categorized into noisy and curated. Several algorithms were proposed to incorporate noisy PK, but none address curated PK. Another challenge is that much of the PK is not stored in databases or not in a unified structured format to be accessible by inference algorithms. Moreover, no GRN inference method exists that supports post-prediction PK.

This thesis addresses those limitations with three solutions: PEAK algorithm for integrating both curated and noisy PK, Online-PEAK for post-prediction interactive feedback, and DeTangle for visualization and navigation of GRNs.

PEAK integrates both curated as well as noisy PK in GRN inference. We introduce a novel method for GRN inference, CurInf, to effectively integrate curated PK, and we use the previous method, Modified Elastic Net, for noisy PK, and we call it NoisInf. Using 100% curated PK, CurInf improves the AUPR accuracy score over NoisInf by 27.3% in synthetic data, 86.5% in *E. coli* data, and 31.1% in *S. cerevisiae* data.

Moreover, we developed an online algorithm, online-PEAK, that enables the biologist to interact with the inference algorithm, PEAK, through a visual interface to add their domain experience about the structure of the GRN as a feedback to the system. We experimentally verified the ability of online-PEAK to achieve incremental accuracy when PK is added by the user, including true and false PK. Even when the noise in PK is 10 times more than true PK, online-PEAK performs better than inference without any PK.

Finally, we present DeTangle, a Web server for interactive GRN prediction and visualization. DeTangle provides a seamless analysis of GRN starting from uploading gene expression, GRN inference, post-prediction feedback using online-PEAK, and visualization and navigation of the predicted GRN. More accurate prediction of GRN can facilitate studying complex molecular interactions, understanding diseases, and aiding drug design.

DeTangle: A Framework for Interactive Prediction and Visualization of Gene Regulatory Networks

Doaa Altarawy

(GENERAL AUDIENCE ABSTRACT)

Proteins are complex molecules that are responsible for most of the functionalities in living organisms. Proteins are produced from genes inside the cells. The production of proteins can be controlled by other proteins causing their production to increase or to decrease. This control is called gene regulation. Studying gene regulation between genes is important because it facilitates understanding diseases and aiding drug design.

In this thesis, I developed computational methods, called PEAK and online-PEAK, to predict the interactions between genes using computer algorithms. My test results show that the method PEAK is more accurate than previous existing methods. I have also built a Web application, DeTangle, to help biologists use the algorithm and visualize the results.

Dedication

This dissertation is dedicated to my husband, Hossameldin Shahin, and to the joy of my life, my twins, Zeyad and Somaia.

Acknowledgments

First of all, all thanks due to ALLAH, may His peace and blessings be upon his prophet, for granting me the chance and strength to complete my PhD.

My gratitude to my advisor, Dr. Lenwood Heath for his inspiration, invaluable guidance, and patience. This work would not have been possible without his encouragement and support.

I would also like to thank my committee members Dr. Ruth Grene, Dr. Cliff Shaffer, Dr. Chris North, and Dr. Mohammed Ismail for their precious feedback, guidance, and efforts reviewing my work and dissertation. It is my honor to have worked and learned from them. I would like to thank Dr. Bert Huang for his useful discussions, Dr. Song Li for all his effort and time in the user study, and Dr. Richard Bonneau for providing me with the source code of Mixed-CLR and the Inferelator.

I express my gratitude to VT-MENA program director, Dr. Sedki Riad for all his efforts and support. He has been and will always be a real father for us. Thanks to the people in the CSA department, Virginia Tech, and the Blacksburg community for making my stay here from the best periods of my life.

Special thanks to my parents who inspired me and supported me with their love and blessings throughout my life. They supported me with inspiration, motivation, and strength. I am sorry that they did not live to see me graduate. I am grateful to my husband Hossam for his unconditional support during my years of study. I thank my twins, Zeyad and Somaia, for being patient with a busy mum.

I also thank all my wonderful friends and sisters who supported me during hard times, especially Nahla Farahat, Sally Hammouda,Fatma-Elzahraa Eid, Dalia, Dina, and Ihsan.

Again, thanks to ALLAH All mighty God for giving me the chance to be where I am now.

This work received support from the VT-MENA program of Egypt and the missions sector of the Egyptian ministry of higher education.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the goals of systems biology is to understand gene regulatory networks (GRNs) and how they respond to perturbations. The problem of GRN inference is to predict the structure of the network using experimental data. Most existing GRN inference methods use gene expression data because of its wide availability; the public database GEO has more than 1.6 million gene expression samples [6]. However, prediction of a GRN from gene expression data remains a challenging problem. In recent years, high-throughput technologies facilitated the availability of different types of biological data, such as gene expression profiles, protein level quantification, ChIP-Chip, and ChIP-Seq. With the abundance of these biological data, computational prediction of GRNs has become more feasible.

Many methods have been proposed to reconstruct a GRN from gene expression data. Common GRN inference approaches include information theory [2, 13, 23, 51], Bayesian networks [60, 65, 88], differential equations [9, 32, 73], regression [33, 84], Boolean networks [37], and Gaussian graphical models [75].

Despite all efforts with dozens of methods for reverse engineering GRNs from gene expression data alone, the prediction accuracy is still low, according to the DREAM consortium [52].

Several methods have been recently proposed to integrate other prior biological information in the prediction of GRNs from gene expression data. Including prior information about the structure of the network was shown to greatly improve the GRN prediction accuracy [31, 50, 73, 81].

Different kinds of biological data are used as prior knowledge for GRN inference, including transcription factor binding sites, Gene Ontology (GO) annotation, functional association, protein-protein interactions (PPIs), and public databases of experimentally verified pathways. Prior knowledge was also used to improve GRN prediction in Gaussian graphical models [75].

We categorize the prior knowledge about the structure of a GRN into two types: curated and noisy. Curated prior knowledge is experimentally verified regulatory relations, which are available in curated pathway databases and in the literature. On the other hand, noisy or inaccurate prior knowledge is any other biological data supporting a possible relation between a pair of genes; however, such prior knowledge does not necessarily imply a regulatory relation. Examples of noisy prior knowledge for GRN inference include TF binding, functional association, PPIs, and GO annotations.

In some curated prior knowledge, even though gene A is known to regulate gene B, their gene expression data may not show a clear causality and thus cannot be incorporated by inference algorithms. This problem is *poor gene expression support* for the prior knowledge. To our knowledge, no method has been proposed to integrate curated prior knowledge from validated biological experiments, especially when the prior knowledge is not well supported by the gene expression data, as will be discussed in Section 2.3.

In this work, we developed an algorithm, PEAK, for integrating both noisy and curated prior knowledge in GRN inference, even with poor gene expression support, which has not been addressed before. PEAK is based on the well-established algorithm Inferelator [9] and extends

the previously proposed method Modified Elastic Net [31], designed for robust integration of noisy prior knowledge, to be able to incorporate both curated and noisy prior knowledge. We propose the novel GRN inference method, CurInf, for curated prior knowledge, and we use NoisInf for noisy prior knowledge.

The GRN is modeled via ordinary differential equations (ODEs), and the machine learning method Elastic Net is used for model selection as in [31]. The prediction algorithm has two phases: coarse-grained and fine-grained. In the coarse-grained phase, mixed context likelihood of relatedness (mixed-CLR) is used to predict potential regulators for each gene [51]. In the fine-grained phase, two modified versions of Elastic Net [25] are used to refine the predictions and to integrate curated and noisy prior knowledge. PEAK makes the following contributions.

- We distinguish between two different categories of prior knowledge that can be used in GRN inference: curated and noisy.
- We developed a method CurInf to effectively integrate curated prior knowledge (experimentally verified regulatory relations).
- We are the first to present a GRN inference method to address the issue of poor gene expression support for curated prior knowledge. In the literature, no method was proposed to integrate such curated prior knowledge.
- We implemented a system that is able to incorporate both noisy and curated prior knowledge in the same model, handling each type differently.

Although much experimentally verified data exists that confirms the regulatory relations between some genes, many of these data are not gathered in structured databases to be accessed by prediction algorithms as prior knowledge. In this way, regulatory network inference algorithms use a limited subset of the existing prior knowledge from the literature

that has an accessible standard format. Given the low accuracy of current computational prediction methods of regulatory networks and the relative limitation of using existing prior knowledge, there is a need for exploring innovative methods to address the interdisciplinary and challenging nature of the GRN inference problem.

In the second part of this thesis, we developed an online algorithm, online-PEAK, that enables the domain expert to interact with the inference algorithm, PEAK, through a visual interface to add their domain experience about the structure of the GRN as a feedback to the system. Submitted feedback from the biologist is considered prior knowledge to the inference algorithm, which was previously shown to improve prediction accuracy [1]. In particular, domain experts are able to add their prior knowledge to a small subset of genes in the visualized regulatory network, or pathways of their expertise, in which they are interested.

With online-PEAK, our system has four contributions.

- First, prior knowledge can be incorporated before, as well as after, GRN prediction. Existing GRN prediction methods allow adding prior knowledge before the prediction, but none enable integrating prior knowledge after the prediction.

- Second, we provide a Web interface for the users to add their post-prediction prior knowledge directly on a visualized GRN rather than in a textual format. Visualizing the predicted network can help the user relate genes and hypothesize new regulatory relations. For instance, they can mark incorrectly predicted regulatory relations based on their knowledge about the incident genes. Moreover, they can add hypothesized relations that they believe might exist.

- Third, we developed an online algorithm, online-PEAK, that can perform in real-time, where users submit their post-prediction feedback about the predicted network and receive the updated GRN in the same session.

- Finally, the underlying inference algorithm, PEAK, is robust to noisy interactions, and noisy prior knowledge will not affect the inference accuracy if they are not supported by the gene expression data.

Another challenging problem in computational biology is the visualization and analysis of biological networks. One of the outstanding problems in this domain is the presentation and navigation of large networks with thousands of genes. Another common problem is the issue of handling multiple pipelines with different input and output formats. In this challenge, the biologists transform their data into the input format of each tool they use. Moreover, they need to transform the output of each tool to match the input of the next tool in the pipeline, which can be a tedious and error prone process. Another reliability concern is the accuracy of the underlying prediction algorithms used within the tool. Finally, some specialized analysis and prediction tools require specific experimental data that is not widely available. Many tools have been proposed in the literature; each has its own advantages and shortcomings.

In this thesis, we present DeTangle, a Web application system and a framework for the inference and visualization of gene regulatory networks. DeTangle is an integrated tool for a seamless analysis of a GRN, starting from uploading gene expression data, GRN inference using PEAK, post-prediction feedback using online-PEAK, and visualization and navigation of the predicted GRN. In addition, DeTangle allows exporting the results in several standard formats. Our system does not require special omics data and utilizes the most abundant type of biological data as the only required input. Most importantly, DeTangle is a framework designed so that it can be easily extended to include additional prediction and analysis algorithms.

Our experiments show that our GRN inference method PEAK utilizes curated prior knowledge to improve the accuracy compared to previous work, even when prior knowledge is not supported by the gene expression data. Using synthetic data and real data from *E. coli* and

S. cerevisiae, our method is able to integrate curated prior knowledge even when it is not supported by the gene expression data. In addition, we show that such poor gene expression support is prominent in real non-synthetic data. For curated prior knowledge, CurInf consistently out-performs NoisInf for any percentage of curated prior knowledge used. For example, compared to NoisInf using 100% curated prior knowledge, CurInf has a 27.3% improvement in accuracy for synthetic data, an 86.5% improvement for *E. coli* data, and a 31.1% improvement for *S. cerevisiae* data.

Furthermore, we tested the ability of online-PEAK to achieve incremental accuracy gain when prior knowledge is added by the user, which simulates the addition of post-prediction prior knowledge interactions by the user. We have tested the incorporation of prior knowledge that may include wrong interactions (noise) with ratios of true:false interactions ranging from 1:1 to 1:10. Our system DeTangle was shown to be robust to noise. Even when the noise in prior knowledge is 10 times more than true prior knowledge, PEAK performs better than inference without any prior knowledge.

I have also gathered user feedback from potential users of DeTangle who are interested in visualization of biological data in the domain of bioinformatics. User feedback was highly positive regarding the usefulness and usability of the system, and users provided me with valuable feature requests. I have implemented most of the requested features and enhancements with a few considered for future work. Finally, I present a case study obtained with collaboration with a biologist in the Department of Crop and Soil Environmental Sciences at Virginia Tech in the *Arabidopsis thaliana* root stem cells pathway.

Chapter 2

Background

2.1 Gene Regulatory Networks

Proteins are one of the major molecular components in biological cells. A protein is produced by the translation of its coding gene after the latter has been transcribed in the cell into messenger RNA (mRNA) by RNA polymerase. The expression of genes is (partly) controlled by proteins called transcription factors (TFs). The expression of a gene can be regulated by more than one TF. TFs bind to the promoter region of the DNA of the target gene and can cause the production of the gene to increase (activation) or to decrease (repression). Target genes themselves can be TFs that regulate other genes. The network that contains gene and TF interactions is called a gene regulatory network (GRN). A GRN is an abstraction of the actual complex process of gene regulation in a biological system. Understanding GRNs, and how they respond to perturbations, is one of the goals of systems biology.

In recent years, high-throughput technologies facilitated the availability of different types and large quantities of biological data, such as gene expression profiles, protein level quantification, ChIP-Chip, and ChIP-Seq. Gene expression profiling measures the activity (the

expression) of thousands of genes simultaneously. Techniques to measure gene expression include microarrays and RNA-Seq. Those techniques measure levels of mRNA transcripts produced by genes as an indicator for the activity of their corresponding proteins. ChIP-Chip and the more recent method ChIP-Seq are technologies used to investigate interactions between TFs and their binding sites in the DNA sequence, thus giving a general indication of potential interactions between TFs and genes.

With the abundance of biological data, computational prediction of GRNs from gene expression data has become more feasible. Such computationally predicted regulations can guide experimental design and validation in various domains. For instance, Moignard et al. [57] used single-cell gene expression with network analysis to build a gene regulatory network model for blood development in a mouse embryo. Subsequently, they were able to experimentally validate and unveil many regulations for Sox and Hox factors. The reverse engineering of a GRN can help in studying complex biological processes, understanding diseases and biomarkers, and designing drugs.

A gene regulatory network can be represented as a directed graph, where nodes are genes and edges are regulatory relations between genes. TFs are represented in the graph by their coding gene; thus there is only one type of node in the graph. The problem of GRN inference is to predict the structure of the graph using experimental data. Most existing GRN inference methods use gene expression data because of its wide availability.

2.2 GRN Inference Methods

Many algorithms have been proposed to predict a GRN from gene expression data. Model-based approaches propose a model for a GRN and then fit the available experimental data to find the parameters that define the structure of the network. Popular GRN approaches in-

clude information theory [2, 13, 23, 51], Bayesian networks [60, 65, 88], differential equations [9, 32, 73], regression [33, 84], Boolean networks [37], and Gaussian graphical models [75].

Information theory-based methods compute a score between every pair of genes to predict the likelihood of their interaction. Scores used include correlation, mutual information, and CLR (context likelihood of relatedness) [51]. Differential equation models define the change in the expression level of a gene as a function of the expression values of its transcription factors (TFs). Although non-linear differential equations are more capable of explaining complex regulatory relations, linear models are more popular due to computational tractability. In Bayesian network (BN) models, expression levels of genes are considered random variables. Edges in a BN represent the conditional dependence of genes (nodes). BN cannot include cycles (feedback loops in the GRN) or support time series data. Dynamic Bayesian networks were proposed to address those drawbacks. Reviews on GRN inference methods can be found in [20, 34, 35, 46, 61, 64, 77, 80], and a recent list of methods in each network modeling approach is available in [49].

Inferring gene regulatory networks from gene expression data is known to be a difficult problem. The Dialog on Reverse Engineering Assessment and Methods (DREAM) is a collaborative work within the research community to find better insights into the growing biological data using new computational methods. DREAM posts systems biology research problems as challenges to the community to solve. Participants submit their solutions and the organizers evaluate and rank their work in publications in top journals such as Science and Nature. In the DREAM5 challenge [52], the best accuracy in GRN prediction was 50% using the average of AUROC score (area under receiver operating characteristic curve) and AUPR score (area under precision and recall). The best prediction was achieved by combining predictions from the top performing methods in the challenge. Despite all efforts with dozens of methods for reverse engineering regulatory networks from gene expression data, the precision is still at most 50%. This literature-wide assessment does not include

methods that use prior knowledge, because information about the network, such as gene names, was not provided to the participants.

2.3 Inference Methods with Prior Knowledge

Numerous methods have been recently proposed to incorporate other prior biological information (i.e., omics data) in the prediction of GRNs from gene expression data. Including prior information about the topology of the network was shown to be a promising strategy in several studies [31, 50, 73, 81].

Different kinds of biological data are used as prior knowledge for GRN inference, including TF binding, Gene Ontology (GO) annotation, functional association, protein-protein interactions (PPIs), and public databases of experimentally verified pathways. Studham et al. [73] used functional association as prior knowledge to infer a GRN. Although functional association is undirected and does not necessarily imply a regulatory relation, using functional association resulted in a slightly improved accuracy in simulated and yeast data. Chen et al. [15] used natural language processing on literature publications to generate a prior network for GRN prediction. Then, they used a genetic algorithm to predict the GRN using gene expression and the constructed prior network. Multiple sources of prior knowledge were integrated with Bayesian networks to infer GRNs in [41, 81].

Adding prior knowledge was also proposed for improving other types of gene networks like association networks. Wang et al. [78] developed a method called prior lasso that adds prior information in the construction of gene association networks. Li et al. [48] proposed weighted graphical lasso to add prior knowledge to the inverse covariance matrix that represents a gene association network.

Isci et al. [41] provided a Web server with a user interface to input different kinds of prior

knowledge as a matrix of zeros and ones and predict the GRN using a new approach called Bayesian network prior. Prior knowledge was also used to improve GRN prediction in Gaussian graphical models [75]. Greenfield et al. [31] developed a method to incorporate prior knowledge about the structure of the network in GRN inference that is robust to false priors.

We categorize the prior knowledge about the structure of a GRN into two types: curated and noisy. Curated prior knowledge is experimentally verified regulatory relations, which are available in curated pathway databases and in the literature. On the other hand, noisy or inaccurate prior knowledge is any other biological data supporting a possible relation between a pair of genes; however, it does not necessarily imply a regulatory relation. Examples of noisy prior knowledge to GRN inference include TF binding, functional association, PPIs, and GO annotations. Note that the notion of noisy prior knowledge does not refer to noise in measurements, but rather to the fact that prior knowledge data such as PPIs does not correspond precisely to a regulatory network. Several methods have been proposed to specifically address the incorporation of such noisy prior knowledge, such as [31].

In some curated prior knowledge, it is possible that a gene A is known to regulate gene B, but their gene expression data may not show a clear causality and subsequently cannot be predicted correctly by inference algorithms. This problem is *poor gene expression support* for the prior knowledge. One of the reasons for such poor support is that gene expression data measures the levels of mRNA transcripts as an estimate of the protein levels. This is not always an accurate estimation, especially in time series data, since the half life of mRNAs can differ from the half life of their proteins. Also, some regulatory relations may be hidden by others due to the complexity of regulatory mechanisms such as post-transcriptional regulation including small RNAs. Other reasons include noise in the gene expression data and experimental design that does not capture all regulatory relations.

To my knowledge, no method has been proposed to integrate curated prior knowledge from validated biological experiments, especially when the prior knowledge is not well supported

by the gene expression data.

In this work, I developed PEAK for integrating both noisy and curated prior knowledge in GRN inference which has not been addressed before. Curated or reliable prior knowledge can come from experimentally verified pathways whereas noisy prior knowledge can be any other supporting information about the network structure such as PPI and TF binding.

2.4 Visualization and Analysis Tools for GRNs

Many tools have been proposed in the literature for the visualization and the analysis of gene networks in systems biology ready for the biologist to use. Every tool addresses a specific problem or need in the community; each has its own advantages and shortcomings. Among the tools for the visualization and the analysis of biological networks are Cytoscape [72], STARNET [44], Navigator [12], Genotet [83], VANTED [43], Cerebral [7], BisoGenet[53], LegumeGRN [76], and GeneMANIA [79].

Web-based tools in particular are increasingly proposed for different biological analysis purposes such as the analysis of regulatory sequences [56], protein functional annotations [17], and visualization of multi-omics data [42].

Needing to handle multiple pipelines each with different tools to process omics data is a common problem in bioinformatics domain. In this challenge, the biologists transform their data into the input format of each tool they use. Moreover, they need to transform the output of each tool to match the input of the next tool in the pipeline, which can be a difficult and a time-consuming task. Furthermore, some specialized analysis and prediction tools require specific experimental data that is not widely available. Reviewing the literature, no tool provides an end-to-end facility of gene regulatory network prediction with a simple and widely used input.

That is, to my knowledge, there is no tool available that provides a usable and easy way for biologists to upload their data, predict a gene regulatory network, visualize the resulting network, and allow user feedback and prior knowledge to be integrated into the prediction algorithm.

Current visualization tools have different objectives. Some tools address small-scale networks (pathways). Pathway editors can include features such as: drawing, editing, simulating, validating, and/or predicting biological networks. Other tools can handle the layout of large-scale networks but they do not provide an easy way to explore the graph, which is still an outstanding challenge in systems biology [27]. Gene expression data from RNA-Seq experiments can result in a network with up to $30,000$ genes. The hair-ball network shape is still presented to the user in an overwhelming way [74]. Some recent widely-used tools, such as GeneMANIA, avoid drawing the whole graph and show only a small user-specified subset of the network. Few tools are specialized in visualizing large networks; however, they do not provide prediction capabilities such as NAViGaTOR [12].

Kharumnuid et al. [45] presented a review on tools for GRN inference and visualization. A review for visualization tools of omics data in general can be found in [27] and tools to explore biological networks in [74]. The following is a review of some available network visualization and prediction tools for systems biology.

NAViGaTOR

NAViGaTOR [12] is a desktop graphing tool (written in Java) for scalable visualization and editing of large biological networks. NAViGaTOR provides fast graph layout algorithms with OpenGL hardware acceleration. Users can upload their network data in text formats such as PSI-MI XML, BioPax, and GML. Moreover, networks can be accessed through supported databases such as the PPI database I2D [11] and Pathway Commons [14].

NAViGaTOR allows direct editing and drawing of nodes and edges in the graph in addition to automatic network layouts. It supports several graph layouts such as circular or linear. NAViGaTOR supports collapsing and expanding grouped nodes and edges selected by the user. Some basic network analysis methods are included such as finding shortest paths, cliques, and hubs. In addition, NAViGaTOR can generate random networks. The edited network can be exported in many formats.

NAViGaTOR is mainly an efficient graph editor and visualization tool, and it does not support any network prediction algorithms.

GeneMANIA

GeneMANIA [79] is a widely-used tool and an example of a recent successful Web-based application in systems biology. GeneMANIA generates a gene function association network inferred by their previously published prediction algorithm [58], using genomics and proteomics data. Next, GeneMANIA visualizes the network to the user with many useful experimentally curated annotations from databases such as GEO [6], BioGRID [10], and Pathway Commons [14]. GeneMANIA currently supports seven organisms [89].

GeneMANIA avoids drawing the whole graph and shows only a selected small part of the network. To obtain a focused view, GeneMANIA requires the user to specify gene ID(s), and it then displays a fixed number of genes in the neighborhood of the selected genes. The resulting network is interactive, where the user can hide/show annotations and view extra information. The visualized network can be exported as a JPEG image, while the network data (such as genes, functions, or interactions) can be saved in a textual format (tab-separated values).

GeneMANIA does not predict gene regulatory networks but rather predicts gene functions from multiple functional association networks using a heuristic based on Ridge regression [58].

Cytoscape

Cytoscape [72] is a powerful desktop application for integrated biological network analysis. Since its first release in 2003, Cytoscape has more than $10,000$ citations [71, 72]. Cytoscape functionalities can be extended using apps (previously called plugins). Networks can be imported from text files or extracted from supported databases for many species. Cytoscape supports both directed and undirected graphs with many automatic layout algorithms.

For years, many tools have been developed as plugins for Cytoscape to utilize its existing interface and capabilities. However, developers faced some challenges to integrate their algorithms with Cytoscape. First, the current supported version is version $3.x$, while version 2 is no longer supported. This can affect previously developed plugins and requires constant updates from developers to upgrade their plugins to the latest platform. Second, Cytoscape is a desktop application, which requires users to download and install the application on their machines. This can be challenging for some applications that require high performance computing or complex backend algorithms with language-specific environments and package set-up. This challenge can be addressed by hosting the application logic and algorithm in a remote server and relying on Cytoscape for visualizing the resulting network. Third, and most importantly, Cytoscape cannot handle large networks efficiently, which can be an issue for problems with large networks such as gene regulatory network inference from gene expression data. Finally, some applications, like our proposed system DeTangle, requires intrinsic changes to the interface. Thus, our tool requires developing a remote server as well as fundamental interface changes. Combined with the inefficient capability of Cytoscape to handle very large graphs, developing our system DeTangle as a Cytoscape plugin was not feasible.

As an extension to the Cytoscape project to handle the issue of large networks, Cytoscape.js [24] was proposed. Cytoscape.js is a JavaScript library for visualizing large networks and it is

currently used in much successful software such as GeneMANIA [79]. I use and extend the Cytoscape.js library for visualizing my predicted GRN in DeTangle.

GRNsight

GRNsight [16] is a Web application that visualizes gene regulatory networks by allowing the user to upload their network file in a Microsoft Excel format (.xlsx). GRNsight uses force layout for automatic presentation of the network and allows the user to change the layout parameters, such as link distance, charge, and gravity to improve the graph layout. GRNsight does not provide any prediction algorithm in the back-end.

D-Map

D-Map [3] has a Web-based interface that allows the user to upload their gene expression data along with seed genes (genes of interest). Next, D-Map predicts the regulatory network by combining different methods from the literature using random walk simulations. The predicted network can then be download as text files. D-Map does not visualize the predicted network and also does not allow using prior knowledge.

cGRNB

cGRNB [82] is also a Web-based server for constructing combinatorial GRNs. A combinatorial GRN integrates data about both TFs as well as microRNA (miRNA) regulators which, interplay in gene regulation [55, 70]. The network has three types of interactions: TF→miRNA, miRNA→target gene, and TF→target gene. The contribution of this server is providing an easy method for the user to upload their data and run a previously published inference method in the server (written in R). However, cGRNB does not visualize the predicted network, and the results are downloaded in CSV (comma-separated values)

file format. The user inputs are special types of expression data: miRNA-perturbated gene expression and parallel miRNA/mRNA expression. The authors expect that those kinds of data will be widely available in the future.

Chapter 3

PEAK: GRN Inference using Prior Knowledge

3.1 Problem Formulation

The problem of computationally inferring a gene regulatory network from gene expression data can be defined as follows. Given N genes with their mRNA expression level measurements for a number of experiments R, it is desired to predict the regulatory relation between each pair of genes. The regulatory relations can be viewed as a directed graph, where the nodes are the genes and the edges represent the regulatory relations between their incident genes. Regulatory genes (source nodes) are called transcription factors (TF) that affect the expression level of their target genes. A regulatory relation between two genes can be an activation or an inhibition. Prior knowledge about the topology of the GRN can be used as an input along with the gene expression data.

We use the formulation proposed by [9] and subsequently followed in [32, 31, 51]. Let $X = (x_1, x_2, \cdots, x_N)^T$ be the observed gene expression levels of N genes in R experiments,

where $x_i \in \mathbb{R}^R$. Gene expression data can come from two types of experiments: time series and steady state. In time series, a perturbation is introduced to the system, and then mRNA expression levels of the genes are measured at consecutive time intervals. For steady state, some perturbation is introduced, but the measurement is done when the system reaches a stable state. The input to the prediction algorithm can be K time series measurements and M steady state measurements, where $K + M = R$.

Time series data can be presented as a matrix X^{ts} with $N \times K$ elements, where $x_i(t_k)$ denotes the measurement of gene i at time k. Hence,

$$X^{ts} = \begin{pmatrix} x_1(t_1) & x_1(t_2) & \cdots & x_1(t_K) \\ x_2(t_1) & x_2(t_2) & \cdots & x_2(t_K) \\ \vdots & \vdots & \ddots & \vdots \\ x_N(t_1) & x_N(t_2) & \cdots & x_N(t_K) \end{pmatrix}. \tag{3.1}$$

Similarly, steady state data can be presented as a matrix X^{ss} with $N \times M$ elements, where $x_i(e_k)$ denotes the measurement of gene i at experiment number k. Hence,

$$X^{ts} = \begin{pmatrix} x_1(e_1) & x_1(e_2) & \cdots & x_1(e_M) \\ x_2(e_1) & x_2(e_2) & \cdots & x_2(e_M) \\ \vdots & \vdots & \ddots & \vdots \\ x_N(e_1) & x_N(e_2) & \cdots & x_N(e_M) \end{pmatrix}. \tag{3.2}$$

All gene expression observations are normalized first using Robust Multichip Average (RMA) [39, 8]. Since gene expression levels vary, it is important to scale the data so that no gene dominates the others in the model selection unless implied by some prior knowledge.

3.2 Core Model

The regulatory relations among genes are modeled as a system of linear ordinary differential equations (ODEs). In the ODE model, the change of the expression level for gene x_i is a linear sum of the expression levels of its TFs [9]. For the sake of clarity, we will describe the model for one gene, and the same procedure should be repeated for all N genes. Let P_i be the set of potential regulators (TFs) of gene x_i (which may include up to all N genes). Let α_i be the first-order degradation rate of x_i, where $\alpha_i = \frac{t_{1/2}}{\ln(2)}$, and $t_{1/2}$ is the half-life of the mRNA of the gene [51, 32, 31]. The ODE of gene x_i can be written as

$$\frac{dx_i}{dt} = -\alpha_i x_i + \sum_{p \in P_i} \beta_{i,p} x_p. \tag{3.3}$$

The $\beta_{i,p}$'s are unknown coefficients. The sign of $\beta_{i,p}$ of a TF x_p shows whether it is an activator (positive) or inhibitor (negative) of gene x_i. If $\beta_{i,p}$ is zero, this means that gene x_p is not a regulator of gene x_i.

3.3 Using the Expression Data

Using the expression data from M time series experiments and K steady state experiments we can build the regression model for each gene x_i. Building the regression model from the expression data presented here is proposed by Bonneau el al. [9] and used in several best performing methods in DREAM3 and DREAM4.

3.3.1 For time series observations

A differential equation can be discretized using finite difference approximation of derivatives. Applying this approximation to equation (3.3) and rearranging the ODE, we obtain

$$\begin{aligned} \frac{x_i(t_{k+1}) - x_i(t_k)}{t_{k+1} - t_k} &= -\alpha_i x_i + \sum_{p \in P_i} \beta_{i,p} x_p(t_k) \\ \tau_i \frac{x_i(t_{k+1}) - x_i(t_k)}{t_{k+1} - t_k} + x_i(t_k) &= \tau_i \sum_{p \in P_i} \beta_{i,p} x_p(t_k) \end{aligned} \tag{3.4}$$

where $\tau_i = \frac{1}{\alpha_i}$, and $x_i(t_k)$ and $x_i(t_{k+1})$ are expression levels measurements of gene x_i at times t_k and t_{k+1} respectively. Here the TFs $x_p(t_k)$ are time-lagged with respect to the target gene $x_i(t_{k+1})$ by one time point. The left hand side of equation (3.4) is considered the response variable of gene x_i, which we rename as a new variable y_i:

$$y_i(t_{k+1}) = \tau_i \frac{x_i(t_{k+1}) - x_i(t_k)}{t_{k+1} - t_k} + x_i(t_k). \tag{3.5}$$

Rewriting equation (3.4) using the response variable y_i, we get

$$y_i(t_{t+1}) = \tau_i \sum_{p \in P_i} \beta_{i,p} x_p(t_k). \tag{3.6}$$

3.3.2 For steady state observations

The measurements are taken when the system is in a stable state, i.e., $dx_i/dt = 0$. This means that the cause and the effect appears in the same measurement e_l without time-lag. Using $dx_i/dt = 0$ in equation (3.3), we get

$$x_i(e_l) = \tau_i \sum_{p \in P_i} \beta_{i,p} x_p(e_l). \tag{3.7}$$

The previous equation can be written using the response variable y_i, where $y_i(e_l) = x_i(e_l)$, to obtain

$$y_i(e_l) = \tau_i \sum_{p \in P_i} \beta_{i,p} x_p(e_l). \tag{3.8}$$

The final model equation for each response gene y_i can be written by combining times-series (equation (3.6)) and steady state (equation (3.8)) to get

$$y_i = \begin{pmatrix} y_i(t_2) \\ \vdots \\ y_i(t_K) \\ \\ y_i(e_1) \\ \vdots \\ y_i(e_M) \end{pmatrix} = \begin{pmatrix} \sum_{p \in P_i} \beta_{i,p} x_p(t_1) \\ \vdots \\ \sum_{p \in P_i} \beta_{i,p} x_p(t_{K-1}) \\ \\ \sum_{p \in P_i} \beta_{i,p} x_p(e_l) \\ \vdots \\ \sum_{p \in P_i} \beta_{i,p} x_p(e_M) \end{pmatrix}. \tag{3.9}$$

Solving Equation (3.9) to find the values of the β's is equivalent to determining the regulators (incoming edges) of gene y_i. The coefficient of each regulator determines its effect on y_i. Positive $\beta_{i,p}$ means x_p is an activator of y_i, while a negative value means it is an inhibitor. If $\beta_{i,p}$ is zero, then x_p is predicted to have no regulatory effect on gene y_i.

The complete regulatory network can be inferred by solving the ODE for each gene and finding its corresponding β's.

3.4 GRN Inference Algorithm

There are two sets of unknown parameters in the final model Equation (3.9): the set of potential regulators P_i and the coefficients $\beta_{i,p}$'s for each gene y_i. As in [51, 31], mixed-CLR

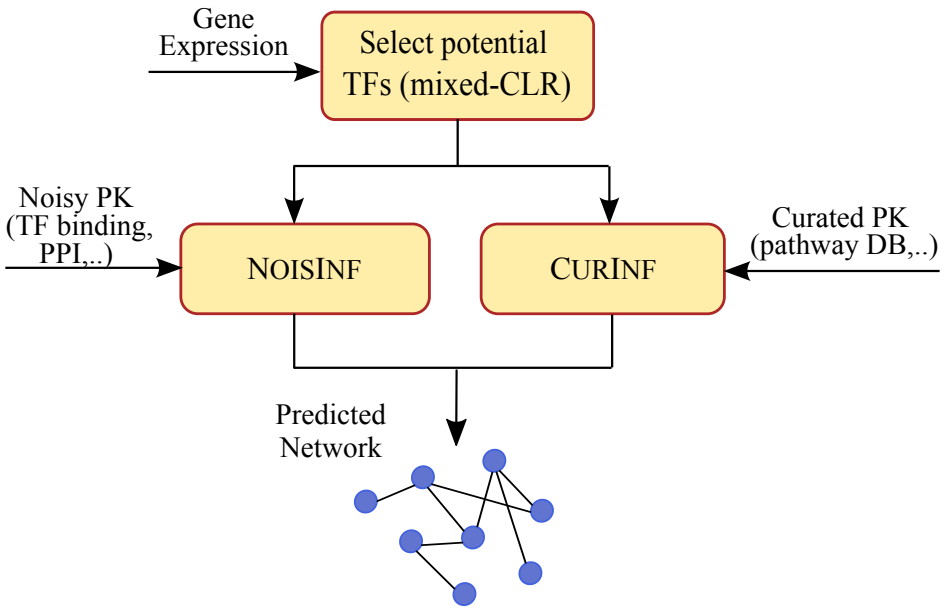


Figure 3.1: PEAK GRN inference system. Gene expression data is fed into the system, along with optionally other data such as TF binding or pathways. Potential TFs are extracted, then based on the prior knowledge either NOISINF or CURINF is applied.

is used as a coarse-grain prediction to find the potential sets of TFs P_i for each response gene y_i. Next, in a fine-grain step, the regression model is solved to find the $\beta_{i,p}$'s using the Inferelator algorithm, which uses elastic net [9, 31]. The pipeline consisting of mixed-CLR and Inferelator was ranked second in the DREAM3 challenge [51]. We integrate prior knowledge into the model using either NOISINF for a noisy prior or CURINF for a reliable prior knowledge in the fine-grain step. PEAK inference system is highlighted in Figure 3.1.

Coarse-grain Prediction Using Mixed-CLR

The purpose of this step is to find a subset of potential regulators P_i for each response gene y_i. Instead of considering all N genes as potential regulators in the regression model for gene y_i, only a small subset is included. Using a constant number of TFs per gene reduces the complexity of the regression model and improves prediction. This filtering is done using mixed-CLR (context likelihood estimator) proposed in [51].

To find the set P_i, first mutual information (MI) is calculated to measure the dependence between every pair of genes. Second, mixed-CLR is used as a background correction method to find the significance of the calculated MI between the TF and its target gene. Finally, for each target gene, the TFs with the highest mixed-CLR scores are chosen as the set of likely regulators P_i, which will be considered in the fine-grain prediction. The cut-off for the top regulators is a parameter, and it was chosen to be 30 TFs per gene as in [31]. The cut-off is a tuning parameter that can be chosen according to the biological relevance of the data, i.e., an estimated upper bound on the number of TFs that are expected to regulate each gene. The following describes how to calculate mixed-CLR.

The mutual information between any two random variables is a measure of their dependence. Let X and Y be two discrete random variables with joint distribution $p(x, y)$ and marginal probability distributions $p(x)$ and $p(y)$. MI is the relative entropy between the joint distribution $p(x, y)$ and the product distribution $p(x)p(y)$, which is defined as

$$MI(X, Y) = \sum_x \sum_y p(x, y) \log \frac{p(x, y)}{p(x)p(y)}.$$

For independent random variables, the MI is zero. In the GRN inference problem, the expression level of each gene is considered a discrete random variable whose values are the observed expression values from times series and steady state experiments. Mutual information is calculated using the smoothing B-spline method described in [18], and it is available as an R package.

For every pair of genes, the mutual information $MI(x_i, x_j)$ is calculated, and the values are stored in an $N \times N$ matrix called M^{stat}. This calculation is repeated using the new response variable y_i, that is finding $MI(y_i, x_i)$ for every pair of response variable (y_i) and design variable (x_i). The resulting matrix is called the dynamic MI matrix M^{dyn}.

The next step is to find the CLR score for every pair of genes from the MI. Given a matrix

M of mutual information, the CLR between two entries x_i and x_j (denoted $z(x_i, x_j)$) is a combination of the z-score of $M_{i,j}$ with respect to the i'th row, and the z-score of $M_{i,j}$ with respect to the j'th column. The z-score of $M_{i,j}$ with respect to the i'th row is given by

$$z_i(x_i, x_j) = \max\left(0, \frac{M_{i,j} - \sum_k M_{i,k}/N}{\sigma_i}\right), \tag{3.10}$$

where σ_i is the standard deviation of the i'th row of M. Similarly, the z-score of $M_{i,j}$ with respect to the j'th column is given by

$$z_j(x_i, x_j) = \max\left(0, \frac{M_{i,j} - \sum_k M_{k,j}/N}{\sigma_j}\right), \tag{3.11}$$

where σ_j is the standard deviation of the j'th column of M. Finally, the CLR score is a combination of both scores

$$z(x_i, x_j) = \sqrt{z_i(x_i, x_j)^2 + z_j(x_i, x_j)^2}. \tag{3.12}$$

Mixed-CLR proposed by [51] does not utilize the exact previous equations, but it uses a combination of M^{stat} and M^{dyn} to find a better background correction. The z-score of $M^{dyn}_{i,j}$ with respect to the i'th row is calculated using equation (3.10) with all values from M^{dyn}. However, for the z-score of $M^{dyn}_{i,j}$ with respect to the j'th column it is given by

$$z_j(x_i, x_j) = \max\left(0, \frac{M^{dyn}_{i,j} - \sum_k M^{stat}_{k,j}/N}{\sigma_j}\right) \tag{3.13}$$

where σ_j is the standard deviation of the j'th column of M^{stat}. Note the use of the values from the M^{stat} in the equation instead of using M^{dyn} for finding the row average. As before, the combined z-score can be calculated using equation (3.12). The resulting matrix is called Z^{mix}, where $z(y_i, x_i) \in Z^{mix}$ is considered a score of the possible interaction between target gene y_i and its TF x_j. Based on that score, the top 30 TFs for each target gene are chosen

to be the set P_i of potential regulators.

Fine-grain Prediction Using Elastic Net

After choosing a set P_i of potential regulators for each gene using mixed-CLR, the next step is to solve the ODE regression model to find the unknown coefficients. We use elastic net to find the final set of predicted regulators for each gene as in [31]. Elastic net is used to solve the regression model for each gene by performing variable selection. After finding the unknown β_i's of the model, a cut-off is applied and a TF with positive β_i is predicted as an activator to the target gene y_i, and it is predicted as an inhibitor if the value of β_i is negative. The magnitude of the coefficient, $|\beta_{i,p}|$, is considered the confidence of the regulatory relation between genes y_i and x_p.

For each gene y_i, it is required to solve the following linear regression model to find the unknown $\beta_{i,p}$'s

$$y_i(r) = \sum_{p \in P_i} \beta_{i,p} x_p(r), \tag{3.14}$$

where r spans all available expression data, including time series and steady state observations. Ordinary least squares (OLS) is a popular method to estimate these $\beta_{i,p}$'s from training data. In general, this is obtained by minimizing the prediction error, which is the sum of squares of the residuals between the predicted value (from Equation (3.14)) and the actual observed value $y_i(r)$. This can be written as an objective function to minimize:

$$E(\beta_i) = \sum_{r=1}^{R} |y_i(r) - f_i(r)|^2. \tag{3.15}$$

In our problem, the function $f_i(r)$ is defined in Equation (3.14), so the previous equation

can be rewritten as

$$E(\beta_i) \quad = \quad \sum_{r=1}^{R} \left| y_i(r) - \sum_{p \in P_i} \beta_{i,p} x_p(r) \right|^2 . \tag{3.16}$$

The estimated $\beta_{i,p}$'s using the OLS method has the limitation of producing a non-sparse solution. In the case of GRN inference, we would like to select a subset of the TFs that regulate the target gene, and expect most of the other $\beta_{i,p}$'s to be zeros, i.e., we want to select a sparse model. Several methods, called regularization methods, exist to find a sparse model, such as lasso, ridge, and elastic net [86]. Elastic net was shown to be more suitable than other regularization methods when a correlation exists between predictors. In a biological context, correlation between the expression levels of TFs is common, and thus elastic net is preferred in solving gene regulation models [86]. Elastic net adds a constraint (or a penalty) $C(\beta_i)$ to the objective function in equation (3.16) to limit the number of variables included in the equation by forcing their sum to be less than a certain value. The elastic net penalty is a combination of L_1-norm and L_2-norm penalties. The elastic net objective function is given by

$$\begin{aligned} J(\beta_i) &= E(\beta_i) + \alpha C(\beta_i), \\ &= \sum_{r=1}^{R} |y_i(r) - \sum_{p \in P_i} \beta_{i,p} x_p(r)|^2 + \alpha C(\beta_i), \end{aligned} \tag{3.17}$$

where $E(\beta_i)$ is the error term, $C(\beta_i)$ is the penalty term, and α is the weight of the penalty term. The error term is given by Equation (3.16), and the penalty term of the elastic net is given by

$$C(\beta_i) = \lambda \sum_{p \in P_i} |\beta_{i,p}| + (1 - \lambda) \sum_{p \in P_i} \beta_{i,p}^2, \tag{3.18}$$

where λ is a balancing parameter between the L_1-norm and L_2-norm penalties. The elastic net parameters α and λ are usually estimated using cross validation.

3.5 Adding Prior Knowledge

We differentiate between two types of prior knowledge: curated and noisy. Regulatory relations from experimentally verified pathways is considered curated or reliable prior knowledge. Noisy or unreliable resources can be derived from PPI networks, physical binding, interactions mapped from homologous genes, or hypothesized relations. We propose a GRN inference method that we call CurInf for reliable prior knowledge, and we use the method Modified Elastic Net (MEN), proposed in [31], to incorporate noisy prior knowledge.

3.5.1 NoisInf for Noisy Prior Knowledge

Greenfield et al. [31] proposed MEN, a robust method for adding noisy prior knowledge to GRN prediction. MEN scales the elastic net penalty to integrate prior knowledge about the topology of the GRN based on adaptive elastic net [87]. In NoisInf, the penalty term in Equation (3.18) is modified by using different degrees of shrinkage on the coefficients $\beta_{i,p}$ for different predictors. This is done by multiplying the L_1term by small values, denoted $\theta_{i,p}$, to prevent shrinkage on the parameter $\beta_{i,p}$ if the corresponding TF x_p is known to be a true regulator of the target gene y_i. Adding prior knowledge in this way tolerates noise and accepts prior information only if it is supported by the gene expression data.

We follow the MEN's main approach [31] for integrating noisy prior knowledge. Each response gene y_i has a prior knowledge vector Θ_i, where $\theta_{i,p}$ equals 1 if no prior knowledge exists between gene y_i and TF x_p and < 1 if prior knowledge exists (i.e., there is a lower constraint on its corresponding coefficient). The weight $\theta_{i,p}$ can be varied according to the amount of confidence in the prior information. The objective function Equation (3.17) in

the case of noisy prior using NoisInf is

$$J(\beta_i) = E(\beta_i) + \alpha\lambda \sum_{p \in P_i} |\theta_{i,p}\beta_{i,p}| + \alpha(1-\lambda) \sum_{p \in P_i} \beta_{i,p}^2. \tag{3.19}$$

3.5.2 CurInf for Reliable Prior Knowledge

NoisInf adds prior knowledge by reducing the penalty term of predictors that are thought to be true TFs of the gene. NoisInf cannot predict a regulatory relation if it is not supported by the gene expression data, since it will be considered noise. In some cases, curated prior knowledge can be unsupported by the gene expression data. For example, if the expression data is noisy or the experimental design did not capture a clear causality relation between the gene and the TF, then the TF will not be predicted as a regulator for that gene even with no penalty term in the objective function Equation (3.19).

We propose a method to add reliable prior knowledge, even if it is not supported by the gene expression data. CurInf adds the prior knowledge to the main error term $E(\beta_i)$ of the objective function Equation (3.17) rather than in the penalty term $C(\beta_i)$. The features of the design matrix of response gene y_i is scaled using the prior knowledge vector Θ_i, where $\theta_{i,p}$ equals 1 if no prior knowledge exists between gene y_i and TF x_p and < 1 if prior knowledge exists. The motivation is that, in regularized linear models, features that have orders of magnitude higher variance can dominate the objective function. Thus, scaling the features causes known TFs to have a relatively higher variance that will favor their selection when solving the model. The objective function Equation (3.17) after CurInf becomes

$$J(\beta_i) = \sum_{r=1}^{R} \left| y_i(r) - \sum_{p \in P_i} \beta_{i,p}\theta_{i,p}^{-1} x_p(r) \right|^2 + \alpha C(\beta_i). \tag{3.20}$$

The term $\theta_{i,p}^{-1} x_p(r)$ means dividing all the expression levels of predictor x_p by the prior

knowledge term $\theta_{i,p}$ that corresponds to its relation with gene y_i. For example, if x_p is known to be a TF for y_i, then $\theta_{i,p}$ will have a small value. This results in up-scaling the term $\theta_{i,p}^{-1} x_p(r)$. If no prior knowledge exists, $\theta_{i,p}$ equals 1, and its expression values $x_p(r)$ are not scaled. We choose to divide by small values $\theta_{i,p}$ instead of multiplying by larger ones to be consistent with the NoisInf notation described earlier.

Coordinate descent [25] is used for an efficient iterative implementation of elastic net, which is proposed in [86]. Cross validation is used to find regularization of the parameters α and λ since they are difficult to choose. CurInf and NoisInf are sensitive to the choice of these parameters, which greatly affects the prediction accuracy. We propose a heuristic to bound the search range of the regularization parameters.

3.6 Bounded Cross-validation for Choosing Model Parameters

CurInf: Since CurInf integrates prior knowledge using the error term $E(\beta_i)$ not the penalty term $C(\beta_i)$ in Equation (3.17), less weight should be given to the penalty term. The penalty term weight α is chosen using cross-validation. We provide a bounded search range for the cross-validation consisting of small numbers (from 0.001 to 0.1) for all data sets.

NoisInf: On the other hand, NoisInf uses the penalty term $C(\beta_i)$ to incorporate prior knowledge; thus it needs more weight for the penalty term to emphasize prior knowledge. In this case, NoisInf needs a range of higher values for α in the cross-validation.

Experimental results show that this strategy is successful in increasing the chance of prior knowledge integration into the selected model, which results in a higher accuracy. Also, based on test results, the balancing parameter between L_1-norm and L_2-norm, λ, has minimal effect compared to α. Thus λ is chosen to be 0.5, giving equal weight to the L_1-norm and the L_2-

norm.

Chapter 4

Online-PEAK

4.1 Motivation

Several methods have been recently proposed to incorporate prior biological information in the prediction of gene regulatory networks from gene expression data. Including prior information about the topology of the regulatory network was shown to be a promising strategy in many studies [1, 31, 50, 73, 81]. Although much experimentally verified data exists that confirms the regulatory relations between some genes, many of these data are not gathered in structured databases to be accessed by prediction algorithms as prior knowledge. It is always up to the developer to choose the appropriate prior data source, while being guided by biologists. In this way, regulatory network inference algorithms use a limited subset of the existing prior knowledge from the literature that has an accessible standard format. Given the low precision of current computational prediction methods of regulatory networks and the relative limitation of using existing prior knowledge, there is a need for exploring innovative methods to address the interdisciplinary and challenging nature of the problem.

To address this, I developed a system, DeTangle, that enables the domain expert to interact with the inference algorithm, online-PEAK, through a visual interface to add their prior knowledge as a feedback to the system. Submitted feedback from the biologist is considered prior knowledge to the inference algorithm, which was previously shown to improve prediction accuracy [1]. In particular, domain experts are able to add their prior knowledge to a small subset of genes in the regulatory network, or pathways, in which they are interested to study. Therefore, existing prior knowledge about regulatory relations between genes that is not stored in accessible databases but is known to the biologists due to their expertise in particular pathways can be integrated into prediction algorithms.

Prior knowledge can be incorporated before, as well as after, the GRN prediction. Post-prediction prior knowledge is added through a Web interface directly on a visualized GRN rather than in a textual format. Visualizing the predicted network can help the user relate genes and hypothesize new regulatory relations. For instance, they can mark incorrectly predicted regulatory relation based on their knowledge about the incident genes. Moreover, they can add hypothesized relations that they believe might exist. Finally, Online-PEAK can perform in real-time, where users submit their post-prediction feedback about the predicted network and receive the updated GRN in the same session.

DeTangle does not require the biologist to associate numeric confidence with the regulatory relations that are impractical to obtain.

Online-PEAK is based on our inference algorithm PEAK [1]. PEAK models the GRN as a system of ordinary differential equations (ODEs) and then uses the machine learning method Elastic Net [86] for feature selection. The prediction algorithm has two phases: coarse-grain and fine-grain prediction. In the coarse-grained phase, mixed context likelihood of relatedness (mixed-CLR) is used to choose a set of potential regulators for each gene [51]. In the fine-grain phase, Elastic net is used to refine the prediction and to find directions in the network. Online-PEAK uses NoisInf, designed for noisy prior knowledge so as to be

tolerant to possible low confidence prior knowledge and hypothesized regulatory relations from the user.

4.2 Interactive GRN prediction Overview

The online-PEAK algorithm uses PEAK as the core model and is implemented in the DeTangle system explained later in Chapter 5. The concept and main idea of interactive GRN prediction can be used with any system that satisfies the online and runtime requirement of the algorithm as will be discussed in this chapter. Thus, other inference methods other than PEAK can potentially be integrated and used if desired. Online-PEAK has three main steps: initial prediction phase, interaction phase, and re-evaluation phase as illustrated in Figure 4.1. Each step is explained below.

4.2.1 Initial Prediction Phase

In this step, an initial prediction of the GRN is generated using the full gene expression data and any pre-prediction prior knowledge provided by the user. The only required input to this phase is time series and/or steady state gene expression data from RNA-Seq or microarray experiments. Optionally, prior knowledge information regarding regulatory relations between genes and a list of potential transcription factors can be supplied. After that, the inference algorithm PEAK is used to predict a GRN from the given gene expression data. The output from this phase is an initial prediction for the regulatory network. The predicted GRN is then visualized and presented to the biologist using the DeTangle system.

This step does not run in realtime, and the processing of large networks (2000 or more genes) can take more than 15 minutes depending on the server settings (see Section 6.8 for runtime estimates on the benchmark data sets). This initial prediction phase is evaluated only once

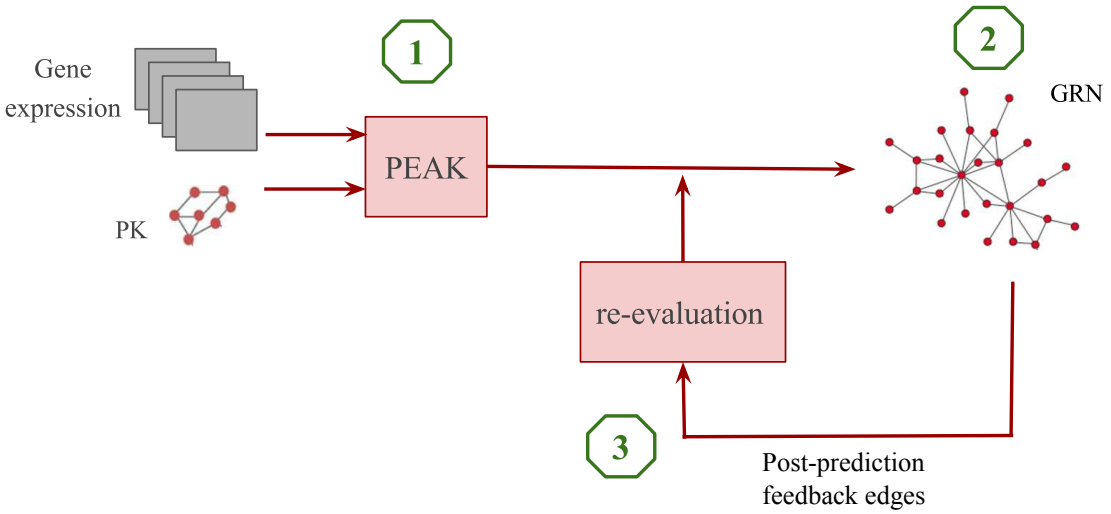


Figure 4.1: Online-PEAK: interactive gene regulatory network inference with post-prediction feedback.

in the beginning of the algorithm.

4.2.2 Interaction Phase

After the initial predicted gene regulatory network is evaluated, the network is visualized to the user using the DeTangle system. The expert user can explore the predicted network and navigate to a pathway of interest to modify. The prediction includes confidence values for each edge. The user can add their feedback to the network in two forms: adding new edges or removing existing predicted edges. Visualizing the predicted GRN can give the biologist an intuition about possible relations based on their experience to add to the network. In addition, they can add regulatory relations from their prior knowledge about the pathway. Expert users can also mark existing edges predicted by the algorithm as unlikely (deleted interactions) based on their knowledge and experience.

The algorithm used by online-PEAK is tolerant to noisy prior knowledge, thus false post-prediction feedback has a low impact on the accuracy of the predicted GRN as shown in the Section 6.4. These post-prediction interactions can represent either prior knowledge or hypotheses of the expert user. The resulting network after user interactions is considered an improved version of the GRN from the expert's point of view.

4.2.3 Re-Evaluation Phase

Next, the user can submit the modified gene regulatory network to the server for re-evaluation. The submitted GRN is used as a feedback input to the online inference algorithm, online-PEAK. We proposed a method to handle both positive and negative prior knowledge. Added edges are considered positive prior knowledge that is integrated by reducing the penalty term $\theta_{i,p}$ in Equation (3.19) as described earlier. Negative prior knowledge means interactions that

are hypothesized to be unlikely or not possible which the user has deleted from the predicted network as post-prediction prior knowledge. Negative prior knowledge is integrated by increasing the penalty term $\theta_{i,p}$ for those transcription factor regulators in Equation (3.19). Online-PEAK inference algorithm uses the submitted network as prior knowledge along with the previously evaluated data from the provided gene expression data to find an improved predicted network. The resulting network is displayed back to the user for further assessment, and the process can be repeated. This re-evaluation phase can run in real-time as will be explained next.

4.2.4 Requirements for the Core Online Algorithm

As mentioned earlier, other prediction algorithms for gene regulatory networks, other than PEAK, can be used as the core method for our online algorithm. In fact, multiple methods can be used or a consensus of several methods. Nevertheless, any such core inference algorithm must satisfy four requirements.

- First, the algorithm must be able to integrate prior knowledge in addition to gene expression data. This capability is essential for the post-prediction prior knowledge to be utilized by the online algorithm, which has been shown to greatly improve the accuracy.
- Second, it should be tolerant to noisy prior knowledge since some of the feedback can be hypothesized regulatory relations between genes. Although this feature is not essential for the online algorithm to be functional, it contributes to its robustness and reliability.
- Third, the inference algorithm should be able to accept an initial prediction to refine and to improve. After the initial network is presented to the user, the post-prediction feedback is used to produce a locally improved sub-network. Partial update of the

network is crucial for any online algorithm to be able to avoid re-evaluation of the whole network.

- Fourth, the refinement (re-evaluation) step must run in a feasible amount of time to be presented to the user in real-time. This is important since our online algorithm runs through a visual interface and the results are displayed directly to the user.

Our proposed inference algorithm, PEAK [1] (Chapter 3), satisfies those requirements and thus is suitable as the core of our system. The PEAK algorithm is based on the methods proposed in [51] (ranked second in the DREAM3 challenge) and [32] (ranked first in the DREAM4 challenge). It also includes the method NoisInf, which is a robust method that is able to integrate noisy prior knowledge about the topology of the regulatory network and is tolerant to that noise. Any other algorithm can be used as the core of the online algorithm if it satisfies the earlier requirements.

We used PEAK to achieve an online-PEAK algorithm that can incorporate post-prediction prior knowledge in real-time through an interactive visualization of the predicted GRN using our Web application, the DeTangle system.

4.3 Online-PEAK Algorithm for GRN Inference

PEAK is used in the core of the online inference algorithm in the initial prediction phase as well as in the re-evaluation phase to produce the predicted GRN network. Re-evaluation of the whole network after user feedback can take a significant amount of time, which is not practical to display in real-time to the biologist. We solved this challenge by partially updating the network based only on the edges that the user modified. Specifically, sink nodes (target genes) of any submitted feedback edges are considered for re-evaluation. Other non-target nodes are not evaluated, and their previously predicted in-coming edges (regulators)

from the initial prediction phase are used.

Prior knowledge is integrated in the fine-grain step (as explained in Section 3.5), thus the coarse-grain step in PEAK (Mixed-CLR) is not affected by this post-prediction prior knowledge. Therefore, the coarse-grain step is evaluated only once in the initial prediction phase, and the resulting potential regulators predicted by this step is stored to be re-used in the re-evaluation phase of the online algorithm.

Pseudocode of the proposed GRN inference algorithm is summarized in Algorithm 1. First, in line 1, in the initial evaluation phase, the design matrices X and Y (which has the adjusted gene expression) along with mixed-CLR clr (which defines the potential regulators of each gene as described in Section 3.4) are evaluated, otherwise; this step is skipped. Thus, the coarse-grain prediction using mixed-CLR is evaluated only once. Next, if the algorithm is called in the re-evaluation phase, the subset of target genes $\hat{Y}$ that were modified in the graph G as post-prediction feedback are extracted. After that, the fine-grain steps (lines 8 to 16) are executed for each gene in $\hat{Y}$. The highest potential TFs for the gene y_i are extracted from clr as well as their corresponding design matrix X_{subset}. Next, the prior knowledge is translated into the vectors: Θ_i^{high} (for curated prior knowledge) and Θ_i^{low} for noisy prior knowledge. Step 12 scales the features in the design matrix for curated prior knowledge as in Equation (3.20). After that, the model selection algorithm, Elastic net, is evaluated to solve the ODE regression model. This is done with scaling the penalty terms in Equation (3.19) with Θ_i^{low} to integrate noisy prior knowledge. Finally, the set of predicted regulators $E(y_i)$ are extracted with their corresponding coefficient values, $Score(i)$, to be reported as the confidence score of each predictors of gene y_i. E is the final adjacency list of the predicted gene regulatory network, and $Score$ defines the edge weights of the network.

Algorithm 1 is re-evaluated each time the user chooses to submit his/her feedback to the server for re-evaluation in the re-evaluation phase in Figure 4.1.

Algorithm 1 Online-PEAK

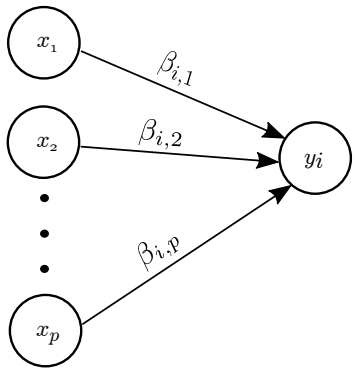
Input: Gene expression data $GeneExpr$ and prior knowledge Graph G

Output: Adjacency list of predicted interactions E, and their corresponding confidence scores $Score$

```
if initial evaluation then
    X, Y := prepare design Matrix and response Matrix using GeneExpr
    clr := calculate mixed-CLR matrix
    Ŷ := Y                                  ▷ all genes are considered in the initial phase
else
    Ŷ := extract modified target genes in G
end if

for gene y_i in Ŷ do
    P_i ← arg max nonzero(clr(i))           ▷ extract indices of predictors with the highest score
    X_subset ← X(P_i)                       ▷ extract gene expr of potential regulators of y_i
    Θ_i^low, Θ_i^hight ← extract noisy and curated prior knowledge of y_i from G
    X_subset ← X_subset / Θ_i^high          ▷ CurInf scaling for curated prior knowledge
    coef ← ElasticnetCV(y_i, X_subset, Θ_i^low)   ▷ solve the regression model using cross
                                                    validation Elastic net with NoisInf
    E(y_i) ← arg nonzero(coef)              ▷ extract indices of regulators with nonzero coef
    Score(i) ← nonzero(coef)                ▷ extract values of nonzero coeff as the final score
end for

return E, Score
```



$$y_i(r) = \sum_{p \in P_i} \beta_{i,p} x_p(r)$$

Figure 4.2: The gene regulation model equation and the corresponding graph representation.

4.3.1 Feasibility for Re-Evaluation at Run-time

In this section, we show that the post-prediction prior knowledge has an effect only on the target genes of the prior knowledge without any effect on the rest of the graph. Moreover, this partial evaluation of the graph can be done in a constant amount of time in practice. This means that the online inference algorithm can be used in an interactive real-time Web application.

In our model, the expression level of each gene in the graph is defined as a differential equation where the rate of change of the gene equals the sum of the expression levels of all incoming edges (regulating genes). As described before in Section 3.3, the equation for gene y_i for a set of r experiments can be reduced to the form:

$$y_i(r) = \sum_{p \in P_i} \beta_{i,p} x_p(r).$$

For each response gene y_i, the set P_i represent the set of its regulating genes. In graph representation, this means for every $x_p \in P_i$, there is an edge from x_p to y_i with weight $\beta_{i,p}$. This is depicted in Figure 4.2.

A regulatory graph is defined as a directed graph $G = (E, V)$, where V are the genes and $e_{ij} \in E$ donates that gene v_i is a regulator (TF) of gene v_j. Let K be a set of prior knowledge

edges, where k_{ij} indicates that v_i is known to be a regulator of gene v_j. Let the set S be all genes that appear as target nodes in K. Post-prediction update of the GRN G has an effect on the incoming edges of the target genes S without any impact on the incoming edges of the rest of the nodes of the graph.

To prove this claim, let us consider a graph G with two genes x_1 and y_2 and no edges. After adding prior knowledge as an edge from x_1 to y_2 ($x_1 \rightarrow y_2$) with weight $\beta_{2,1}$, the equation for x_1 has no change. The equation of the target node y_2 will change to include x_1, that is $y_2(r) = \beta_{2,1}x_1(r)$. Assuming that the theorem holds for a graph G with $n-1$ edges. Adding a prior knowledge edge to G from gene x_s to y_t will change the equation of the target gene y_t to be

$$y_t(r) = \beta_{t,s}x_s(r) + \sum_{p \in P_t} \beta_{t,p}x_p(r).$$

For all other nodes in the graph, since the incoming edges for each node did not change, their equation (the sum of incoming edges) will not change. Thus the only change in the model occurs to the target node of the added edge. Similarly, if the prior knowledge is a change in edge weight, only that coefficient in the equation of the target node will change.

Finally we used the heuristic tested in Section 6.8 that limits the maximum number of potential regulators P_i to a fixed number. Thus, the running time of the online algorithm is bounded by the number of affected target genes multiplied by the number of potential regulators (constant). In practice, the number of modified genes by the user using the Web interface is limited to a small subset of the graph that is visible to the user in each submitted iteration phase. Therefore, in practice, the algorithm can be evaluated in real-time.

Chapter 5

DeTangle

5.1 Introduction

DeTangle is a Web application that provides interactive visualization of a gene regulatory network (GRN) and an online GRN prediction algorithm. DeTangle brings together the top performing computational inference methods with the biologist's prior knowledge and experience to achieve more accurate predictions (as shown in Section 6.3). With a visualization of the predicted GRN, our Web interface enables biologists to interactively assess the output by allowing them to modify the edges of the graph. Expert biologists can add or confirm information about the network structure based on their knowledge and experience. They can also add hypothesized interactions after they inspect the visualization of the predicted GRN network. Some of the hypotheses and intuitions can emerge after inspecting the visualized GRN and would be difficult to infer from raw gene expression data or textually formatted GRN predictions. User interactions are considered prior knowledge that can be submitted to the online prediction algorithm on the server. The predicted network is re-evaluated and then presented back to the user to assess again.

Post-prediction prior knowledge has two advantages over using only pre-prediction prior knowledge. First, visualizing the network can give the biologist an intuition about possible relations based on experience. This is not an easy task without visually inspecting the predicted network. Second, the user can mark incorrectly inferred relations based on knowledge and experience, giving low weight to unlikely edges. An overview of the DeTangle system and some of its use cases is explained next.

5.2 Overview and Use Cases of DeTangle

DeTangle, as shown in Figure 5.1, consists of two components: the inference system and the visualization system. The inference system runs on the server, while the visualization runs on the client (the user's browser).

The user input to the system is gene expression data (transcriptomic data) from microarrays or RNA-seq experiments. Also, the user can optionally upload a list of potential transcription factors and any other prior knowledge about the structure of the network such as verified pathways of protein-protein interactions (input data is explained in Section 5.3.1). After the data is submitted to the sever, the method PEAK, described in Chapter 3, is used to predict an initial gene regularity network as follows. First, mixed-CLR is calculated to select potential regulators from the gene expression data. Next, model selection is performed to fine tune the prediction and to find direction in the network. NoisInf is used for noisy prior knowledge while CurInf can be used for curated prior knowledge [1]. After that, the network is visualized to the user with the ability to navigate the network through several interactions. After reaching a subgraph of interest, the biologist can add their prior knowledge to the graph by adding edges or marking edges as deleted.

Finally, the user can submit the modified network to the online algorithm in the server for

re-evaluation. The feedback from the user interactions is fed into the online algorithm using NoisInf and can be performed in real-time as described in Section 4.3.1. DeTangle also has many capabilities to navigate the network, view gene expression data, and to save and load the visualized data.

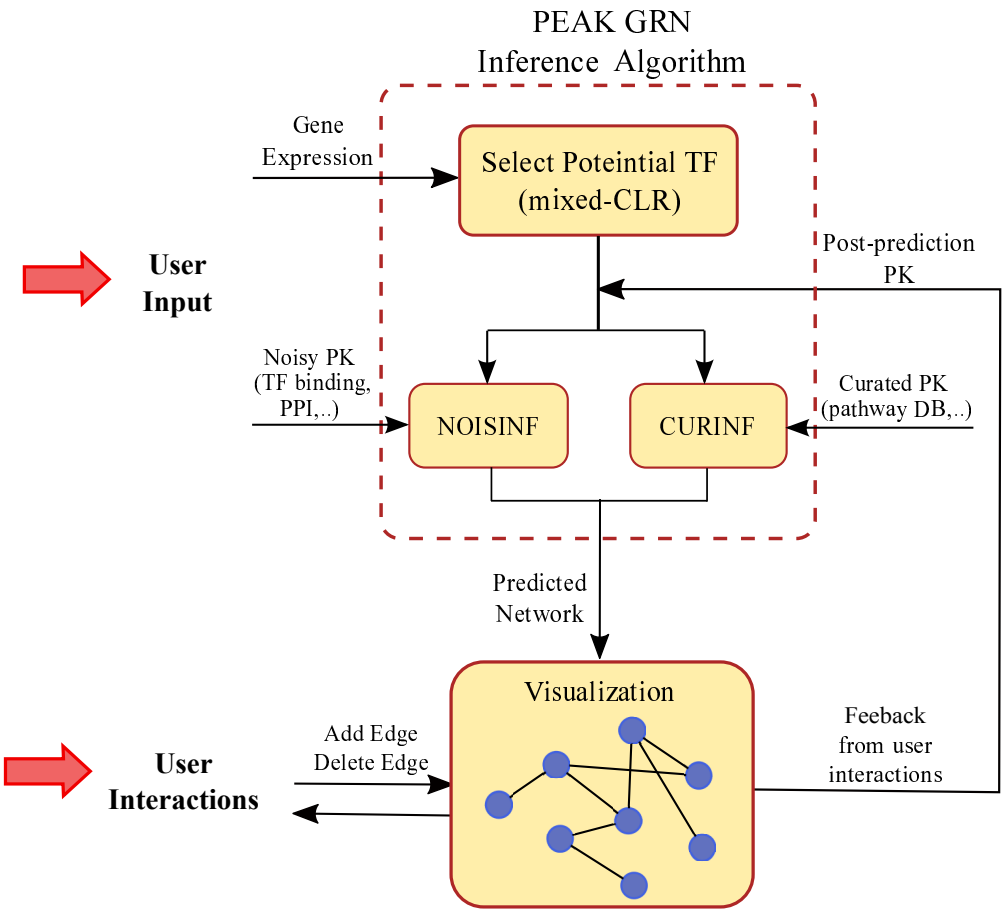


Figure 5.1: DeTangle: Prediction and visualization of gene regulatory network with user feedback as prior knowledge.

5.3 Functional Components

This section describes the functional components of the DeTangle system, including: input and output data, network layouts, interactions and their types, and loading and saving the

results.

5.3.1 Input Data

Four types of inputs can be uploaded to DeTangle to predict a gene regulatory network using the backend algorithm PEAK, each of which are explained below. Only transcriptomic data are required.

Transcriptomic Data

Transcriptomic data, or gene expression data, are the only required input to DeTangle. Transcriptomic data from either DNA microarrays or from RNA-Seq (RNA sequencing) can be used. Gene expression measures the expression levels of mRNAs transcripts in a cell. Multiple samples can be included from time-series or steady state experiments. Also it is possible to include several replicates for the same experiment. Optionally, details about the gene expression data can be provided using the metadata file. Details about the gene expression file format can be found in Appendix A.1.

Experimental Metadata

Experiments Metadata is a file describing more information about each experiment or sample in the gene expression data. The metadata about an experiment includes whether it is part of a time series (and at what interval samples are spaced) or it comes from a steady state measurement when the system reached a stable state. Also information about perturbations and repetition can be included. The file format for the expression meta data is described in detail in Appendix A.2.

Potential Transcription Factors

Despite being optional, providing a list of potential transcript factors can greatly improve the accuracy by reducing the search space for potential regulators. The file lists the names of the genes that can possibly be regulators or transcription factors of other genes. This list should be comprehensive, since only those genes will be considered as regulators. If no list of transcription factors is provided, all genes will be considered for the search space in the underlying algorithm PEAK.

Prior Knowledge

The user can also provide prior knowledge when uploading their data before the initial prediction of the gene regulatory network. This prior knowledge is used in the NoisInf algorithm in PEAK to account for possible inaccurate or non-reliable interactions, which do not necessarily correspond to regulatory relations such as protein-protein interactions and TF-binding. The input file lists pairs of regulator-target genes with a sign showing whether the relation is an activation or an inhibition. Adding prior knowledge was shown to greatly improve the accuracy of the predicted gene regulatory network [1, 31, 50, 73, 81]. Prior knowledge can also be added after the initial prediction into the resulting graph after the prediction is performed.

5.3.2 Output

DeTangle has two main views to be presented to the biologist: the predicted gene regulatory network and the parallel plot of the gene expression data.

Gene Regulatory Network

Figure 5.3 presents the predicted gene regulatory network tab for a demo network from *Arabidopsis thaliana* root stem cells explained in Section 7.2. Nodes of the graph represent genes or gene products (such as proteins) and edges are regulatory relation between their incident genes. Two types of edges exist in the graph: activators and inhibitors. Nodes and edges are selectable, which will be shown in bold black. For example, in Figure 5.2, gene *wox5* and the edge between genes *mgp* and *shr* are selected. In addition, nodes can be "highlighted", which is indicated with red circles. Highlighting is an extra feature to tag genes of interest for navigating and searching neighbors, as will be explained in Section 5.3.4. Edges colored in blue are newly added edges by the user, while edges in dashed red are edges that are marked as deleted by the user (as prior knowledge). Edges are rendered with transparency proportional to their prediction confidence by the inference algorithm PEAK. Finally, the network can be dragged to achieve a zoom-in and zoom-out capabilities.

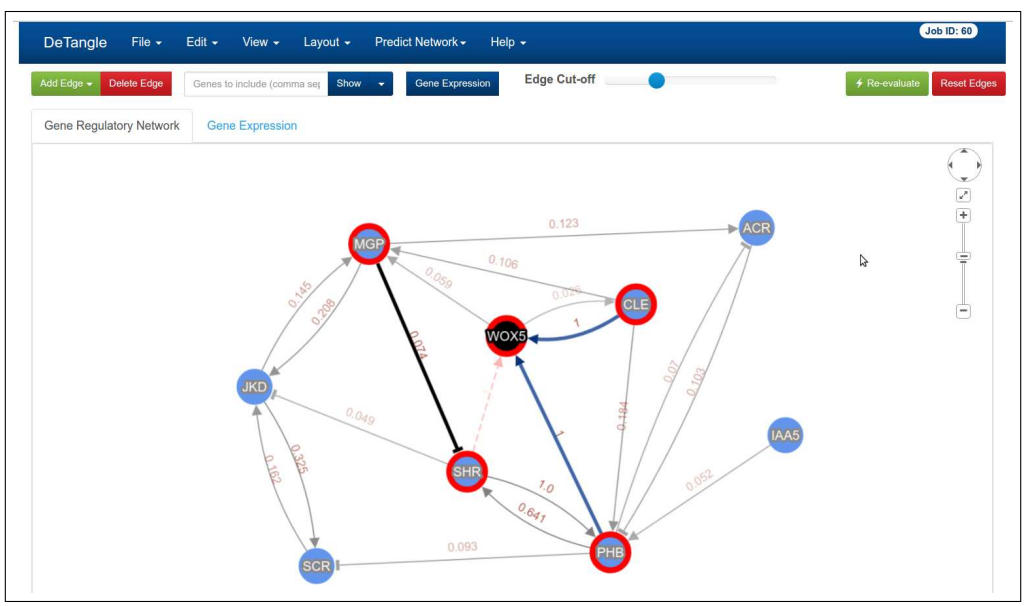


Figure 5.2: Visualization of the predicted gene regulatory network using PEAK as presented by DeTangle system.

Gene Expression Parallel Plot

A parallel plot of the gene expression data is shown in the "Gene Expression" tab, which is the second main view of DeTangle (see Figure 5.3). Parallel plots, or parallel coordinates are a common representation of time-series data and gene expression data [27, 68, 85]. The x-axis is the gene expression number, and the y-axis is the expression values.

In DeTangle, the gene expression is shown only for genes that are "highlighted" (shown in red) in the Network view. This is to reduce number of genes to the desired group of genes under study and to avoid unreadable dense plots. Rendering the parallel plot is refreshed by clicking the "Gene Expression" button. In Figure 5.3, genes on the left can be selected/unselected to show/hide the gene from the parallel plot.

Figure 5.3: Parallel plot of the gene expression data for highlighted genes

5.3.3 Network Layouts

DeTangle uses several automatic network layout algorithms proposed and used in the literature [5, 21, 22, 28, 29, 30]. After the network is initially predicted using PEAK, the nodes have no locations and an automatic layout algorithm is need to present the graph for the first time. After the graph is presented to the biologist, he/she can change node locations

and save it to the server for later retrieval. Figure 5.4 shows an example of available layout algorithms, which are: CoSE-Biknet, CoSE, Cola, Springly, and Arbor. Properties for each layout algorithm can be modified if desired, with default values provided.

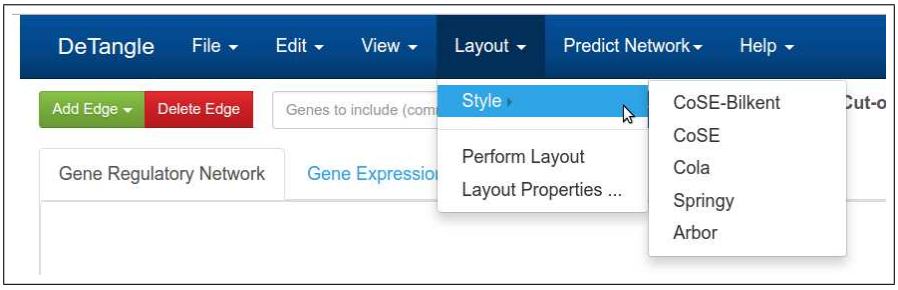


Figure 5.4: Several network layouts are included [21].

5.3.4 Interactions

DeTangle provides many user interactions with the system. We can categorize user interactions with the visualized graph into three type: GRN prediction using PEAK, Online-PEAK interactions, and Network Navigation interactions.

GRN prediction using PEAK

DeTangle provides the biologists with an easy to use method to predict gene regulatory network using their own experiment data for any studied species. The required data, gene expression, is widely available, and DeTangle does not require special kinds of omics data to perform its predictions. The server provides sample files and detailed description of each input format to the user through the interface. Figure 5.5 shows the input form needed to submit a prediction job to the server. The user is then given a unique job id to retrieve their results later. When the user loads their completed jobs, DeTangle visualizes the resulting predicted network and allows the user to download the results in several formats. This capability does not require any computational expertise from the biologist and helps automate

the pipeline of predicting gene regulatory networks.

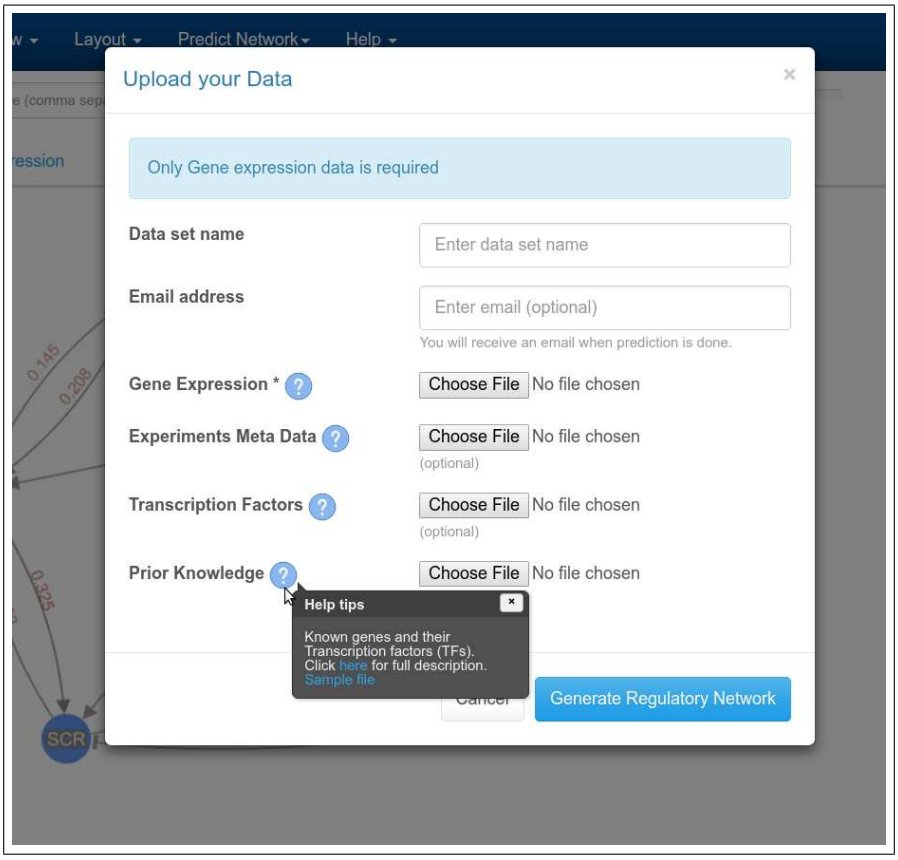


Figure 5.5: Biologists can upload their own data to predict a gene regulatory network.

Online-PEAK

Online-PEAK uses four types of interactions with the predicted network: adding edges, marking edges as deleted, re-evaluation, and resetting added prior knowledge.

Adding and deleting edges are considered post-prediction prior knowledge. The user can add activator and inhibitor edges. An edge can be added to the network by choosing "Add Edge" button then selecting the edge type; activator or inhibitor (as shown in Figure 5.6. Next, the user selects the source gene (transcription factor) then the target gene as highlighted in Figure 5.7). Added edges are colored in a bold-blue color. An edge can be deleted by first

selecting the edge then choosing "Delete Edge" button. Edges marked as deleted are shown in dashed red with lower transparency. Edges are not completely hidden from the graph so that the user knows which edges are marked as deleted and which will be sent as prior knowledge to the online-PEAK algorithm upon choosing re-evaluate.

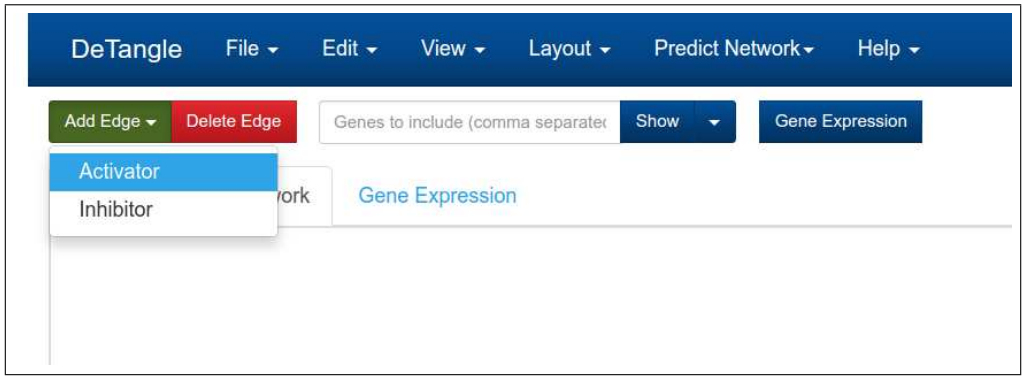
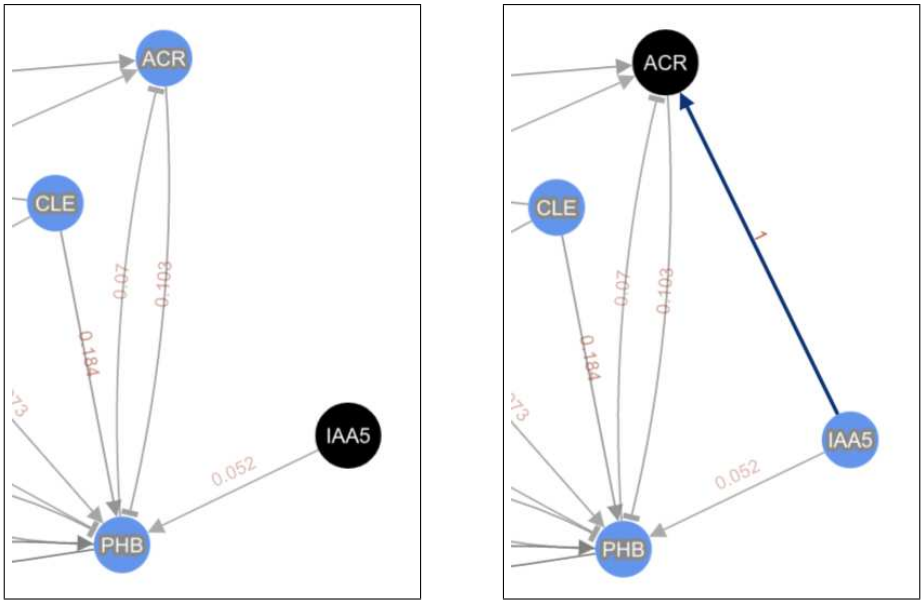


Figure 5.6: Adding an edge (an activator or an inhibitor).



(a) Select source gene (b) Select target gene

Figure 5.7: Adding an edge between IAA5 and ACR by selecting the source gene then the target gene.

Re-evaluation of the network can be done after adding prior knowledge (adding or deleting edges) by the user. This can be achieved by simply clicking the "Re-evaluate" button.

An immediate message will be shown to the user after the re-evaluation of the network is completed as shown in Figure 5.8. As mentioned earlier, this can be done in real-time and the results are shown immediately. Figure 5.9 shows the graph before and after re-evaluation for WOX5 in the *A. thaliana* root stem cells using online-PEAK. In Figure 5.9 (a), prior knowledge is included by adding the edges CLE-WOX5 and PHB-WOX5 and marking the edge SHR-WOX5 as deleted. The resulting network after re-evaluation is shown in (b) with regulatory relation of WOX5 (the only affected target node) highlighted in green.

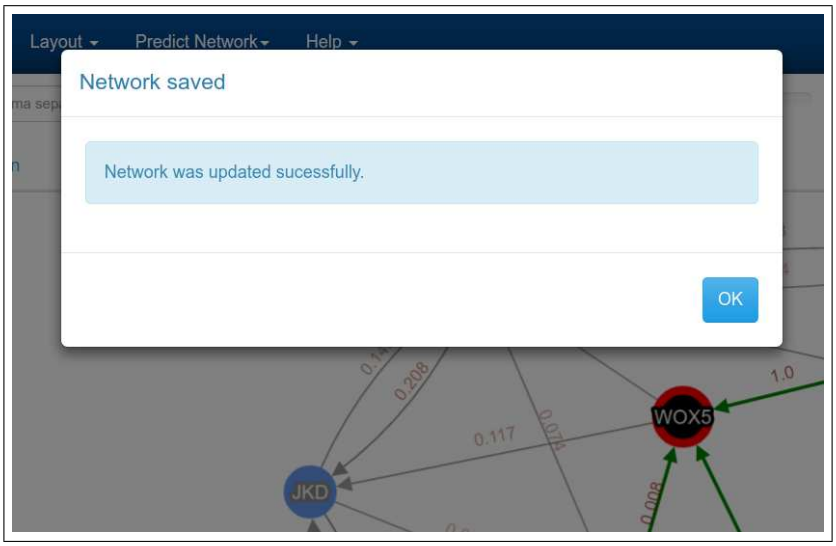
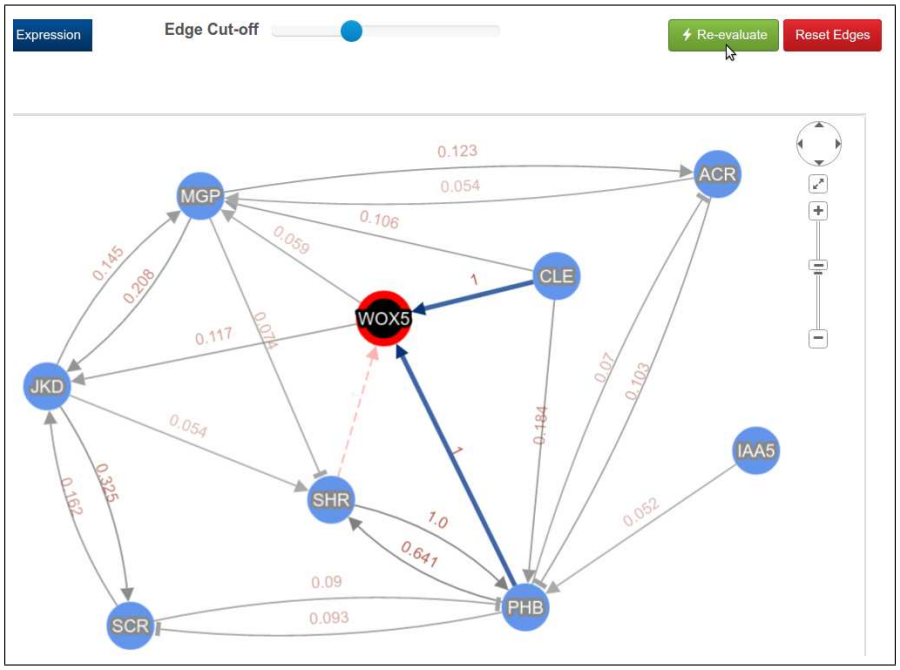


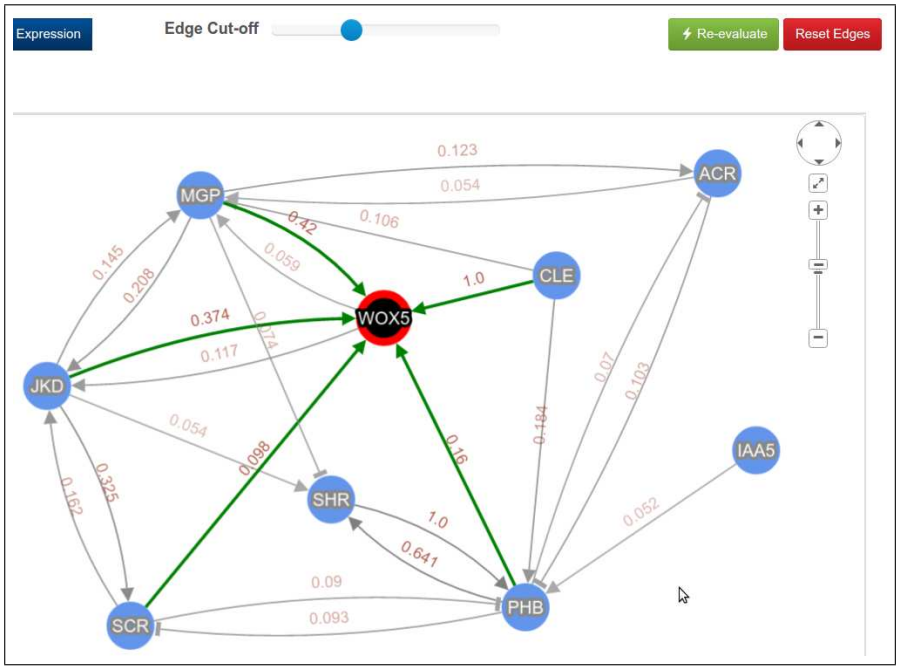
Figure 5.8: Network re-evaluation in real-time.

Network Navigation

The user can navigate and explore the network in variety of ways to reach a pathway or a small network of interest. The goal of network navigation is to reach a small subset of the graph that is of an interest to the biologist. Currently, DeTangle provides some basic filtering and search capabilities. We propose a more complex navigation and dynamic query system in the future work (see Chapter 8). Network interactions and navigations are explained in more detail in Section 5.4.



(a) Prior knowledge added to the network



(b) Network after re-evaluation

Figure 5.9: Re-evaluation of the network for WOX5 using online-PEAK. In (a), prior knowledge is included by adding the edges CLE-WOX5 and PHB-WOX5 and marking the edge SHR-WOX5 as deleted. The resulting network after re-evaluation is shown in (b) with regulatory relation of WOX5 highlighted in green.

5.3.5 Saving Results

The resulting gene regulatory network can be saved in several formats. First, the modified network can be saved directly in the server. Second, it can be downloaded as a JSON file with cytoscape.js representation [24], as a graphXML file [36], or as a PNG image. Moreover, the gene expression parallel plot can also be downloaded as a PNG, a JPEG, a PDF, or an SVG.

5.3.6 Loading/Retrieving Networks

The user can load a job by using its job id. Job ids are unique numeric identifiers used by DeTangle and they are generated by the back-end server. Figure 5.10 shows the dialog to load a job by id. Optionally, the user can choose to load the top k predicted regulatory relations based on their predicted confidence score.

Additionally, predicted gene regulatory network demos are provided for a quick overview of the system under the file menu. The samples provided are from *A. thaliana* root stem cells described in Section 7.2. The jobs have finished execution and their network is ready for viewing immediately. The user can save their own copy of the sample networks to the server as a new job to edit and re-evaluate.

5.4 Network Interactions and Navigations

Hide/Show Genes

Users can hide selected genes or edges from the current visible graph (by right-click and choose hide). This does not delete the genes entirely, but only helps reduce the graph clutter by excluding irrelevant genes from the current pathway of interest.

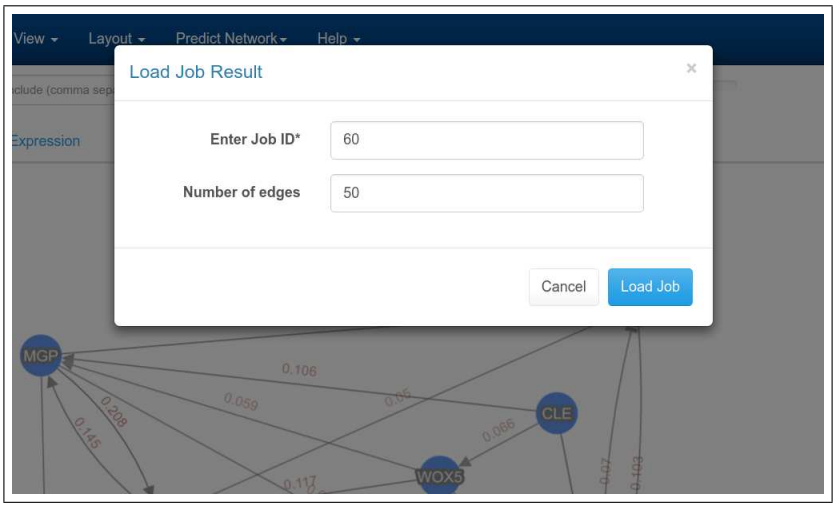


Figure 5.10: Loading saved network data can be done by providing the job id. Optionally, number of top k edges can be used to retrieve the top ranked interactions.

Filtering of Edges

Visible edges can be filtered by their edge confidence. Edge confidence, or weight, is the confidence associated with predicted regulatory interaction between its incident genes. Edges with a threshold below a cut-off are not shown in the graph. The slide bar in Figure 5.11 can be dragged to adjust the edge cut-off threshold value. The cut-off range is on a logarithmic scale since many of the predicted interactions have confidence values in the lower range. This logarithmic scale helps fine tune the desired cut-off to view higher confidence interactions about a given threshold, which is usually on the low side. Figure 5.12 shows the network for *A. thaliana* root stem cell before and after filtering the edges with confidence cut-off of 0.05.

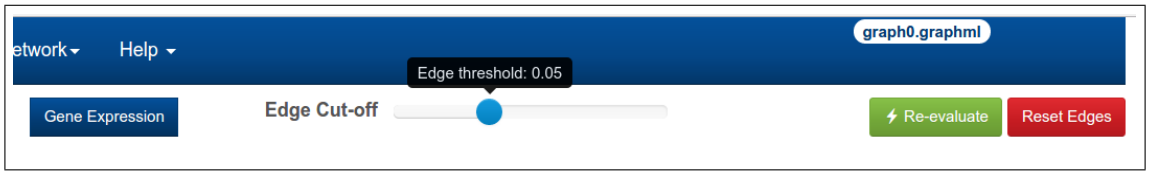
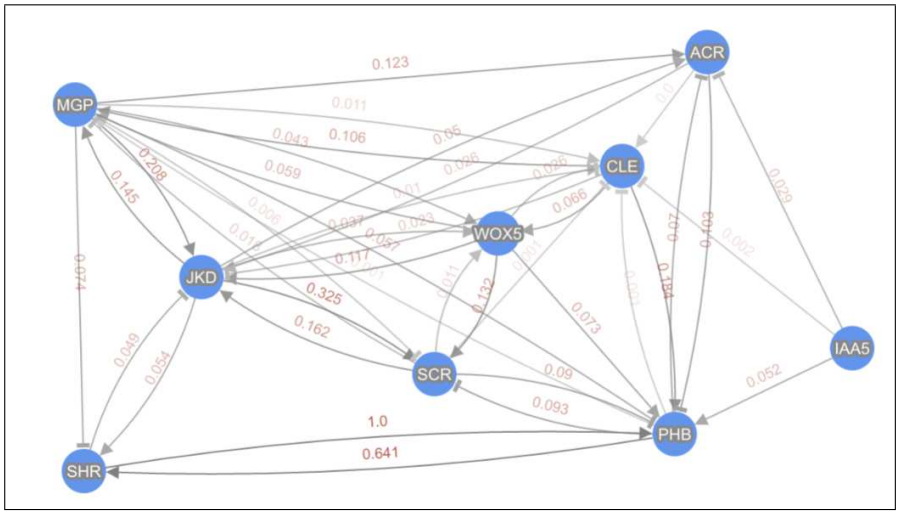
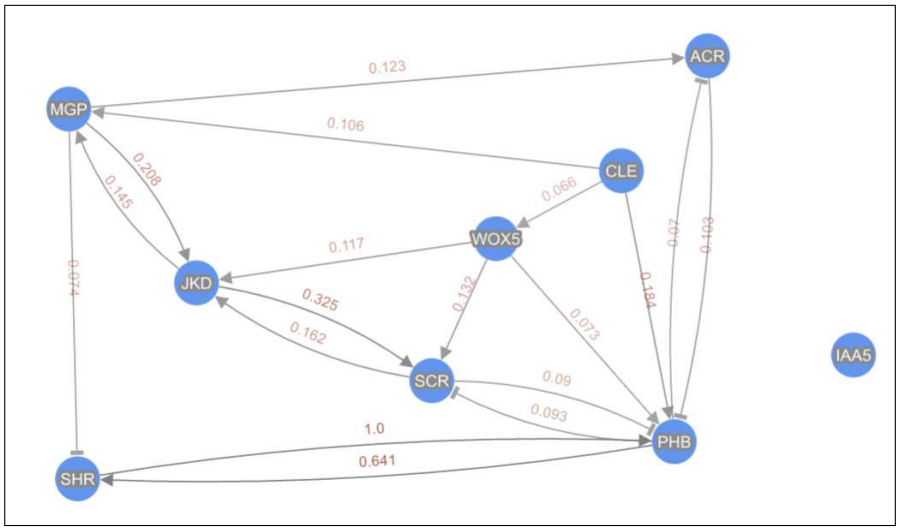


Figure 5.11: Slide bar for changing edge confidence cut-off.



(a) Before



(b) After

Figure 5.12: Using the edge confidence slide bar to filter out edges below the chosen threshold. (a) is the network before edge cut-off filtering, and (b) is the same network after edge cut-off set to 0.05 using the slide bar.

Highlight Neighbors

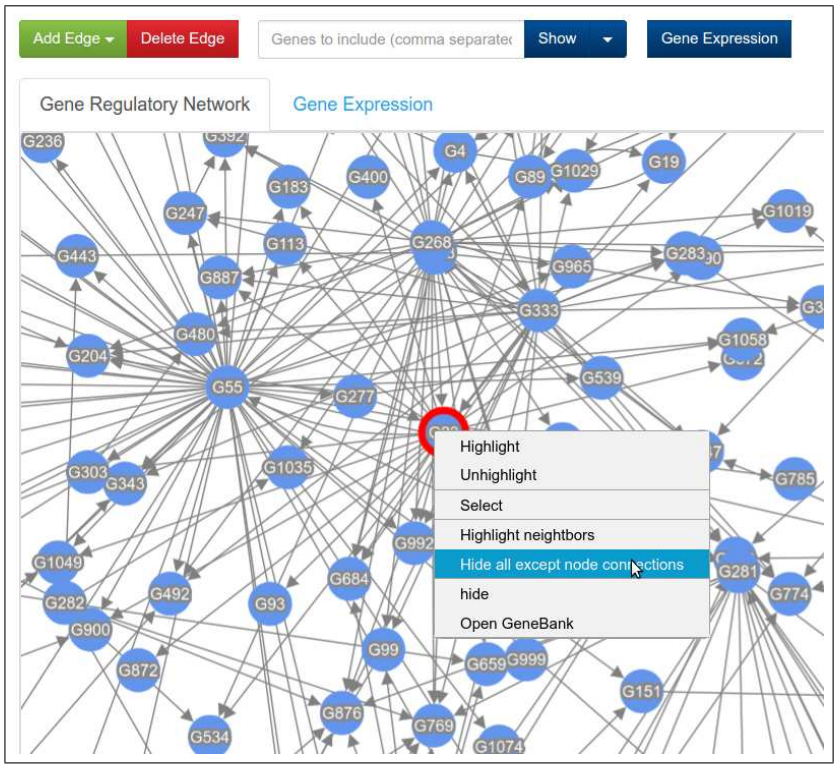
Neighbors of a gene are any genes that have any immediate regulatory relation with that gene, either as a regulator or as a target. Highlighting genes marks them as genes of interest for further filtering or for showing their gene expression. Also, it is possible to hide all genes in the network except highlighted genes to be easy to view genes under study.

Search Genes

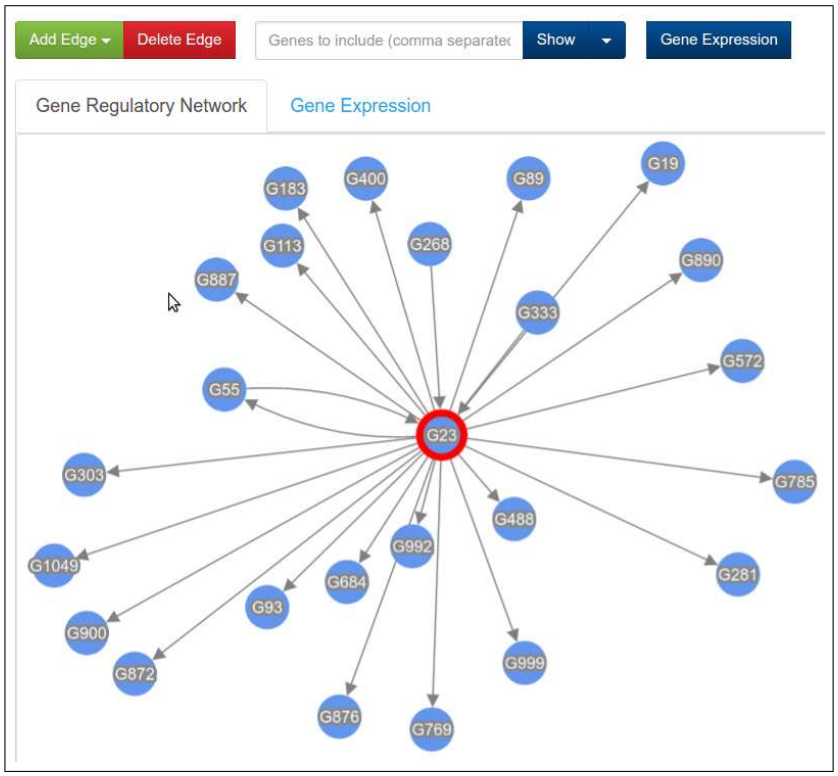
This feature enables the user to search for one or more genes using the search bar. The user can provide a search query in the form of a list of gene names separated by a comma. DeTangle searches for similar gene names using subsequence search, ignoring letter case, and searching within gene names. The resulting nodes are highlighted in red on the network.

Hide all Except Node Connections

To be able to reduce the network into a smaller subset of genes of interest, non-relevant genes along with their regulatory relations can be hidden from the graph. One way to achieve this is to show only the immediate connections of the current selected node (either regulators or targets of the current gene) and hide the rest of the graph. This is useful to focus the view on a specific gene and view all its interactions. The usefulness of this capability is shown in Figure 5.13 for a sample graph from the DREAM5 challenge benchmark [52]. The network in (a) is dense, and the gene of interest, $G23$, is highlighted. In (b), only the immediate connections of the highlighted gene are shown, which improved the visibility of the graph. This can be done for several genes to reach the desired subnetwork.



(a) Before



(b) After

Figure 5.13: A gene under study can be highlighted to show its neighbors and hide the rest of the graph for better navigation of a dense network into a smaller subset of genes of interest.

Hide all except highlighted

It is possible to perform similar task to Section 5.4 but for multiple genes. This can be done by highlighting multiple genes then selecting "Hide all except highlighted". Only immediate connections of highlighted genes are presented, and the rest of the graph is hidden. Combined with the search queries feature, this can reduce the network into genes of interest to the biologist.

Show/Hide Edge confidence

Edge confidence can be shown or hidden by toggling 'Hide edge confidence' in the View menu. This can help the user clear the graph when inspecting the edges confidence values is not needed.

5.5 System Design Architecture

A high-level overview of DeTangle system's architecture is shown in Figure 5.14. The system consists of server-side and client-side components. The server and the client interact using REST (Representational State Transfer) API (Application Programming Interface) calls. The API calls use HTTP POST and GET requests.

DeTangle is designed using a Model-View-Controller (MVC) architecture. MVC is widely used in Web applications and in software system design in general. In an MVC design, the system consists of three separated conceptual components: a model, a view, and a controller. The model is the data layer and the business logic of the application. The view is the presentation layer for viewing and interacting with the data. The controller interacts with both the model and the view. MVC provides many advantages such as low coupling,

efficient design, ease of maintainability, and code reuse.

The client (our system view) consists of the visualization of the gene regulatory network and the parallel plots of the gene expression data and many interactions and navigation capabilities. On the server side, the controller interacts with the client to serve the API calls. The controller is part of the Flask Web framework written in Python. Also, the controller can access any available prediction algorithms on the server, which is currently PEAK and online-PEAK. The model part of Flask reads and writes from and to the database and the file storage.

With this modular design, other visualization software can utilize our server component through the API calls to execute the PEAK algorithm and save and load the resulting GRN. Also the visualization component can use different prediction algorithms if they provide similar APIs to the ones used in DeTangle.

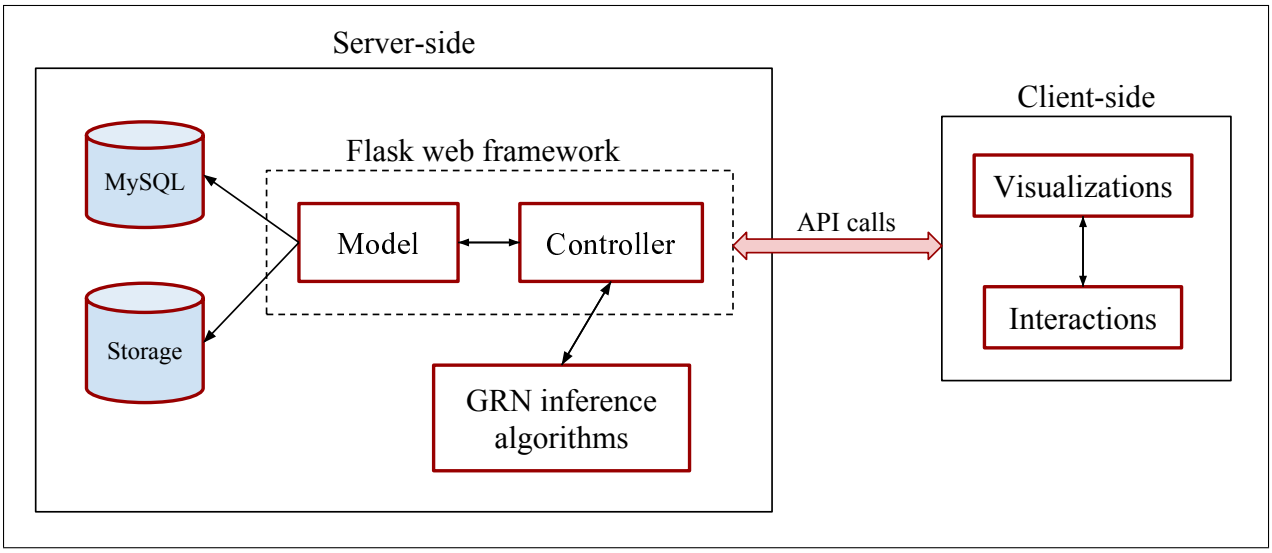


Figure 5.14: Overview of DeTangle system architecture

5.6 DeTangle API Calls

DeTangle uses API calls for communication between the server and the client. There are four main API calls in the system (shown in Figure 5.15): load job, save network, predict GRN, and re-evaluate prediction. Those APIs can be used by other systems to run the prediction algorithms in the server.

The first API is predicting a GRN using the algorithm PEAK. The POST requests includes the input data files uploaded by the user described in Section 5.3.1. The server processes the data and runs the PEAK algorithm asynchronously and also saves the current state of the job in the database. The returned result from this API call is the job id, a unique numerical identifier for the submitted task.

The second API is loading a job by its ID. The GET request should include a valid job id that was previously submitted to the server. DeTangle checks the status of the job in the database and returns a status message to the request. If the job finished execution, the server returns the predicted gene regulatory network (with no node positions) in a JSON format. The network can be visualized by the client side of DeTangle.

The third API is saving the modified network. In this POST request, the client sends the Cytoscape.js network in a JSON format along with its job ID. The network data include gene locations and any other interaction by the user such as added or deleted edges, or highlighted genes. The server saves the network of the given job id to the server.

The fourth API is a re-evaluation of a predicted GRN using online-PEAK. This is a POST request that includes the job id and any added prior knowledge. prior knowledge is sent as two lists: the list of added edges and the list of edges marked as deleted. The server receives the request and retrieves the previously saved results and data of the given job. After that, online-PEAK is called with the appropriate data for the sink nodes of the prior knowledge

as described in Section 4.3. This evaluation can be performed in real-time, and the resulting predicted subgraph is returned to the client which visualizes the network and highlighting the changed edges.

DeTangle does not require registration or login to use. Users in the system are anonymous to satisfy publication requirements. Providing an email address while uploading a job is optional, but if provided, it can be used in the future to notify users when their running job is completed.

5.7 Implementation

Many tools and technologies are used to implement DeTangle. On the server, the Python Web framework Flask is used for the controller and API calls of the server. We use Apache as the Web application server. PEAK and online-PEAK are implemented in Python and scientific computing libraries Scikit-learn [63], Pandas, Numpy, and NetworkX. In the client-side, Cytoscape.js [24], including several plugins, are used for the gene regulatory network presentation, along with JavaScript, HTML, CSS, JQuery, and Bootstrap. Highcharts.js is used for plotting the parallel plot of the gene expression data. MySQL is used as the backend database. DeTangle is tested and deployed on Ubuntu and CentOS operating systems.

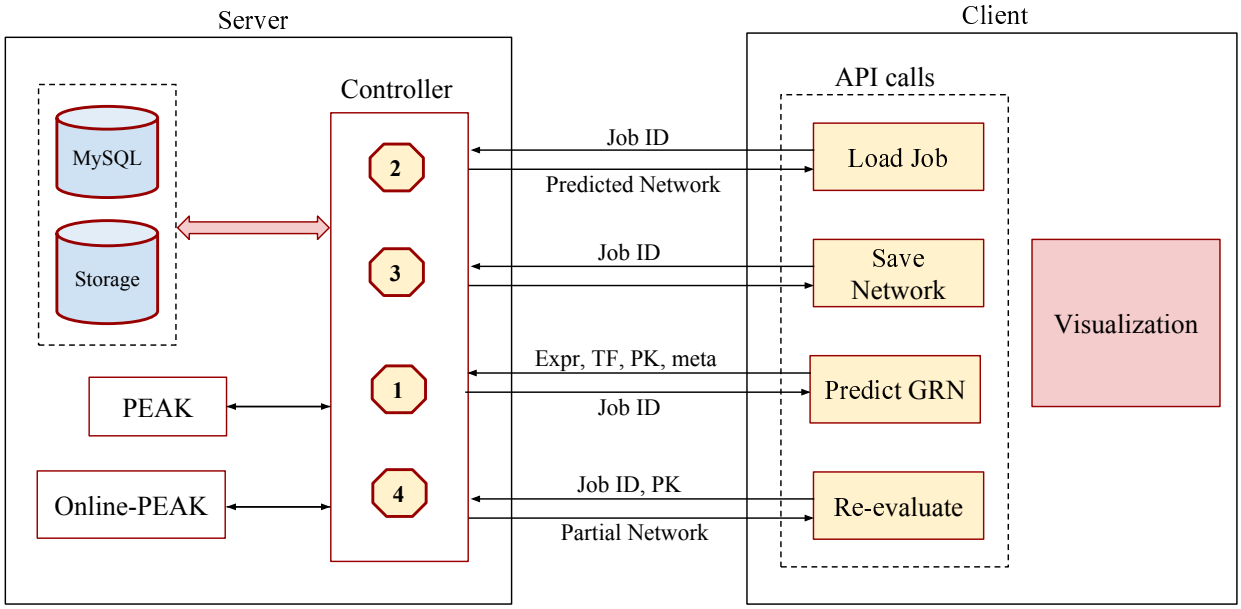


Figure 5.15: DeTangle API calls between the server and the client

Chapter 6

Results

6.1 Data Sets

For testing and analysis of the GRN inference using prior knowledge, we used three data sets from the DREAM5 benchmark [52], summarized in Table **??**. The chosen benchmark has gold standard GRNs (transcription factor-target interactions) available for each data set in order to validate predicted networks. Although predicted interactions not in the gold standard are considered false positives, many can be true interactions, since the gold standard is stringent and incomplete. Each data set consists of gene expression data for N genes in R experiments. Each experiment has a description whether it is time series or steady state, time (if applicable), and experiment repetition information. A list of candidate transcription factors is also provided in the benchmark for each data set to specify which genes are potential transcription factors.

Table **??** lists the data sets used in our experiments. The first data set is a synthetic network generated by the DREAM5 organizers using GeneNetWeaver [69]. The network consists of $1,643$ genes, 195 transcription factors, and 4012 interactions in the gold standard. The

second data set comes from the prokaryotic model organism *Escherichia coli*, which has a well studied GRN. The raw expression data was downloaded from the GEO database. The gold standard GRN was extracted from the curated database RegulonDB version 7 (Platform ID: GPL199) [26] with 805 microarrays, $4,297$ genes, 296 transcription factors, and 2066 interactions in the gold standard. We use a subset of the *E. coli* genes in which each gene has at least one interaction in the gold standard, resulting in $1,100$ genes and 178 transcription factors. The third data set comes from the eukaryotic model organism *Saccharomyces cerevisiae*. The number of microarrays in this data set is 536, extracted from GEO. It includes $5,950$ genes, 333 transcription factors, and $3,940$ interactions in the gold standard. Normalization of the microarray data was done using robust multichip averaging [8]. The fourth data set is a small network from the root stem cells niche in *Arabidopsis thaliana* presented by Azpeitia et al. [4]. This data set is used in the case study in Section 7.2.

Dataset	Samples	Genes	Transcription factors	Edges in gold standard
In silico (DREAM5)	805	1643	195	4012
E. coli	805	1100	178	2066
S. cerevisiae	536	5950	333	3940
A. thaliana	21	9	9	26

Table 6.1: Data sets used for training and testing

We use area under precision-recall curve (AUPR) as the measure of performance as adapted in [31, 52]. AUPR is more suitable than the receiver operating characteristic (AUROC) in applications where the number of positive and negative samples is unbalanced. In GRNs, the network is expected to be sparse, i.e., non-edges are significantly higher in number than existing edges.

6.2 Accuracy of GRN Inference Methods

The core prediction method used in our inference system is a pipeline of the mixed-CLR algorithm and the Inferelator algorithm. We compare the prediction accuracy of this pipeline with other top performing methods in the DREAM5 challenge benchmarking of GRN inference. The pipeline of mixed-CLR and the Inferelator has a comparable performance to the top performing methods in the DREAM5 challenge, GENIE3 [40], and TIGRESS [33], as shown in Figure 6.1. Although GENIE3 and TIGRESS have slightly better accuracy than the pipeline of mixed-CLR and the Inferelator, the later is more suitable for our purpose of integrating prior knowledge.

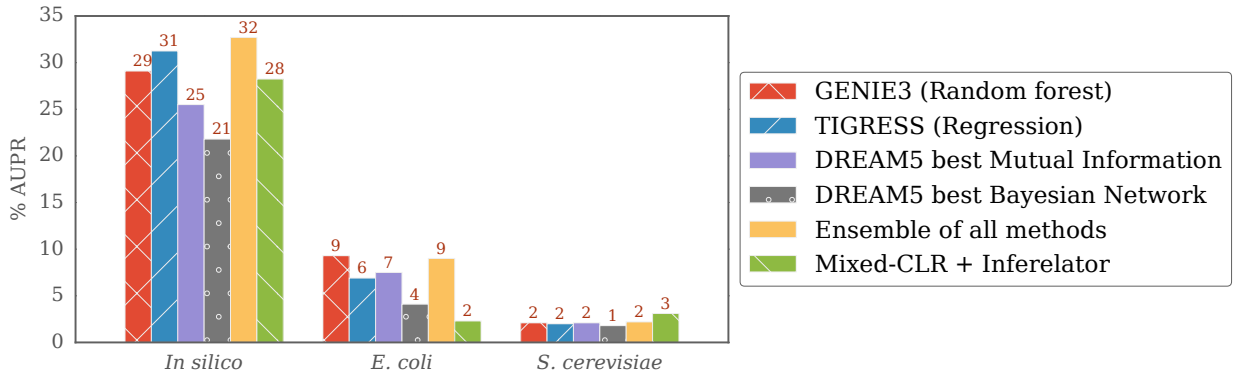


Figure 6.1: Comparison of the AUPR accuracy of the top performing GRN inference methods in the DREAM5 challenge and benchmark. All methods did not use any prior knowledge other than gene expression data. Methods in the figure are: GENIE3 [40], TIGRESS [33], top Mutual Information method [23], an ensemble of several methods [52], and mixed-CLR with the Inferelator pipeline [32].

6.3 Better Integration of Curated Prior Knowledge

We compared the ability of CurInf and NoisInf to incorporate curated prior knowledge within the inference algorithm. Incremental percentages of the gold standard edges are added to NoisInf and CurInf as curated prior knowledge, then the AUPR is calculated

for the reconstructed GRNs. Figure 6.2 shows the accuracy measure AUPR for NoisInf and CurInf on the three data sets. In both methods, the accuracy increases as the prior knowledge percentage increases, which is significantly higher than any method without prior knowledge tested in the DREAM benchmark.

CurInf is superior to NoisInf in the integration of curated prior knowledge resulting in higher accuracy GRN, as shown in Figure 6.2. The difference is more prominent in the *E. coli* and *S. cerevisiae* data compared to the synthetic data. When adding 100% of the gold standard as curated prior knowledge, CurInf was able to improve the AUPR score over NoisInf by 27.3% in synthetic data, 86.5% in *E. coli* data, and 31.1% in *S. cerevisiae* data.

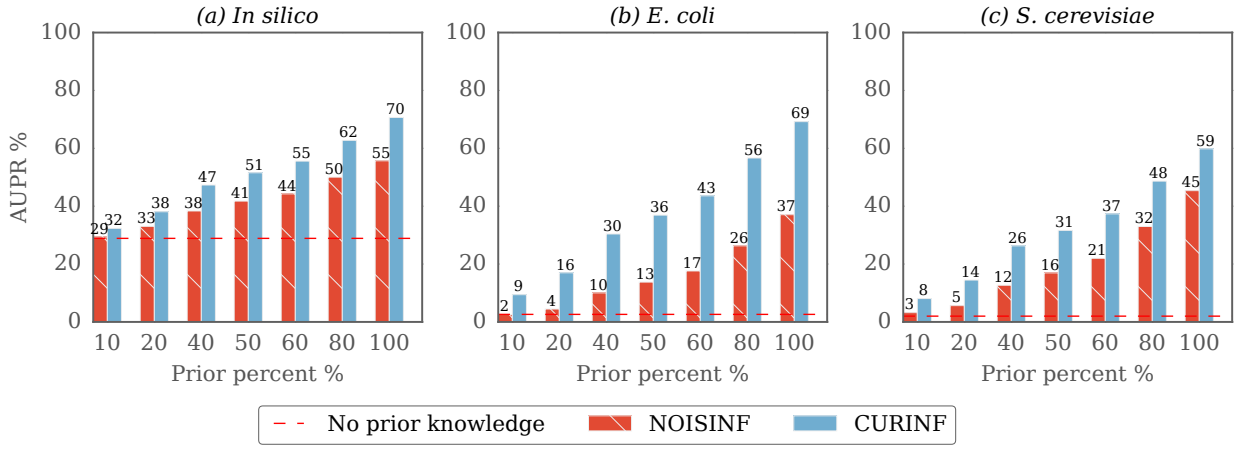


Figure 6.2: The ability of NoisInf and CurInf inference methods to incorporate curated prior knowledge. Different percentages of the gold standard edges are used as prior knowledge to the inference algorithm, and the resulting networks are evaluated. The dotted lines represent the accuracy without using any prior knowledge

6.4 Tolerance to Noisy Prior Knowledge

PEAK uses NoisInf scaling to integrate prior knowledge when it is believed to be noisy. We tested adding noisy prior knowledge to CurInf and NoisInf, and, as expected, NoisInf is more robust and thus more suitable to use with noisy prior knowledge. In Figure 6.3,

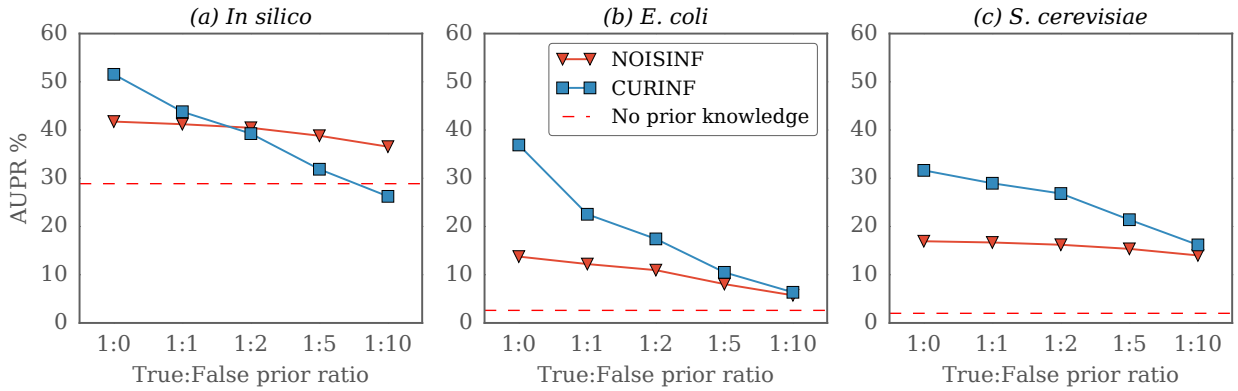


Figure 6.3: Tolerance to noisy prior knowledge. Different true:false prior knowledge ratios are used as input to NoisInf and CurInf, and then the accuracy AUPR is evaluated. As expected, NoisInf is more robust to noise than CurInf.

we added different ratios of true to false prior knowledge. True prior knowledge consists of 50% of the gold standard, while false prior knowledge consists of randomly generated edges between genes and transcription factors that do not exist in the gold standard. Even with 10 times more noisy than accurate prior knowledge, NoisInf performs better than inference without any prior knowledge.

6.5 Gene Expression Support for the Gold Standard

Here we investigate the signal present in each data set and how it is related to the ability to reconstruct the GRN and to integrate prior knowledge. The core model used in CurInf and NoisInf assumes some degree of correlation between the mRNA levels of a transcription factor and a gene for an interaction to be predicted. For time-series expression data, the model uses time lagged relations, meaning that the mRNA level of a gene at time t should be related to the mRNA level of the transcription factor at time $t-1$. For steady state experiments, the system is assumed to reach equilibrium and thus the transcription factor-target relation is considered without time lag. We calculated the transcription factor-target

correlations for the gold standard interactions of each data set as well as the correlation of the unknown interactions. The resulting histograms are shown in Figure 6.4. For the *in silico* data, the distribution of the gold standard edges shows a clear distinction from the unknown edges with more values greater than 0.5 and less than -0.5, making it easier for most inference algorithms to predict. For the real data sets from *E. coli* and *S. cerevisiae*, the gene expression data does not show clear support for all the gold standard, which made those two data sets difficult for the 35 inference methods evaluated with this benchmark [52]. As shown in Figure 6.2, CurInf was able to integrate curated prior knowledge more effectively than NoisInf in those two data sets that do not have strong support for the gold standard GRN.

6.6 PEAK Sensitivity to Parameters

6.6.1 Effect of Prior Weight Parameter

We investigated the effect of the prior weight parameter on the accuracy of the predicted GRN. Different values of the weight parameter θ_i (in Equations (3.19) and (3.20)) were tested in both NoisInf and CurInf (see Figure 6.5). For simplicity, we use the same value of θ_i for all genes in each experiment.

All the gold standard was used as prior knowledge. A prior weight of 1 means no weight is given to the prior knowledge, while smaller values increase the prior knowledge effect. We found that a prior weight of 0.01 or less was enough to produce the best accuracy in our test data. A similar effect was found in [31] when testing the MEN method, where they had a slight peak at a prior weight of 0.01.

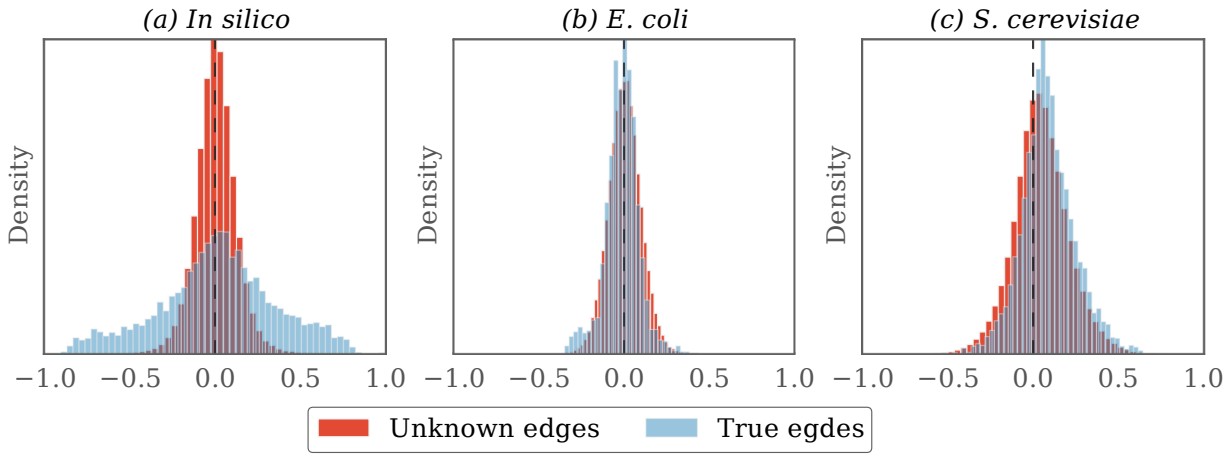


Figure 6.4: Signal in each gene expression data set. For each data set, we plot two distributions to compare: (i) unknown edges: the correlation between the expression level of all genes and transcription factors, and (ii) true edges: the correlation between transcription factor-target pairs present in the gold standard GRN. The two distributions in the *in silico* data have a clear distinction with more correlations greater than 0.5 and less than −0.5 for the gold standard. For the real data sets from *E. coli* and *S. cerevisiae* the gene expression data does not show clear support for all the gold standard which made those two data sets difficult to predict their GRN. CurInf was able to achieve higher accuracy in those real data sets than NoisInf.

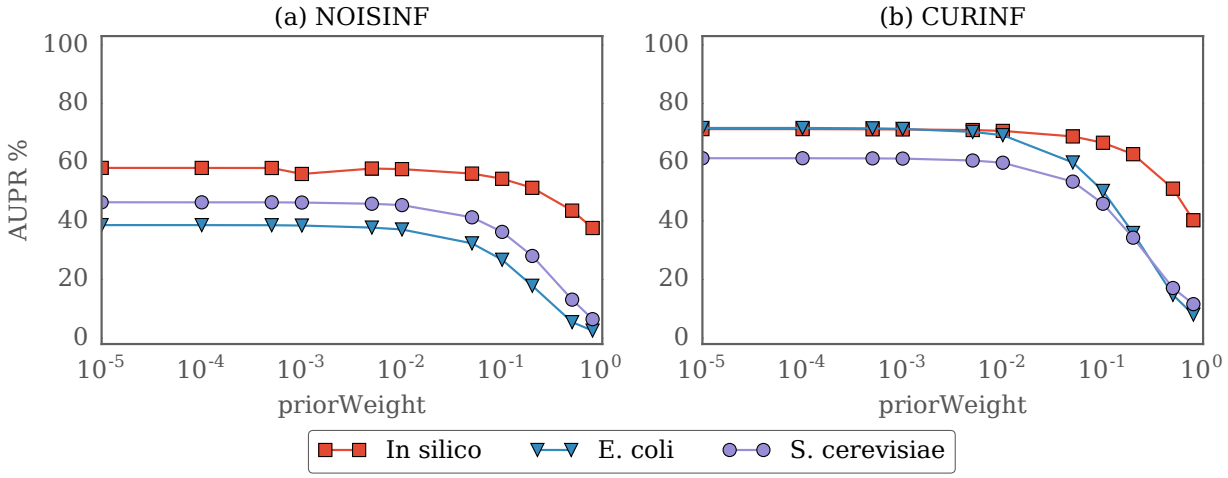


Figure 6.5: Effect of varying prior weight on the accuracy measure AUPR in NoisInf and CurInf inference algorithm when incorporating curated prior knowledge.

6.6.2 Effect of Fitting the Regression Model with an Intercept

We investigate the effect of fitting an intercept of the regression model in the elastic net of our core method on the AURP accuracy and on the ease of setting the model parameters. In general, fitting an intercept is needed if the input features are not centered. We found no effect for intercept fitting on the accuracy in DREAM5 synthetic data, since it is already centered. We also examined whether fitting an intercept has an effect when combined with scaling the features, and there was also no effect of fitting an intercept compared to scaling.

6.6.3 Scaling the Features

Scaling is dividing each feature (transcription factor) by its variance to produce a design matrix with all features having the same variance between 0 and 1. In the DREAM5 synthetic data, the features have comparable variance (with mean 0.341 and variance 0.00044). In the *E. coli* data, the features have greater variance (with mean 1.35 and variance 0.0155). Even small differences in the variance between the features affected the choice of the parameters α (the weight of the penalty term) and lasso term ratio λ defined in Equation (3.17) and Equation (3.18). It is easier to choose regularization parameters when the features are scaled. It can be seen in Figure 6.6 that scaling the features produces a small improvement in the accuracy for most values of α. In the rest of the analysis, we used scaled features to have variance between 0 and 1.

6.6.4 Sensitivity to Elastic Net Parameters

The ability of NoisInf to add prior knowledge depends on the weight given to the lasso term. The lasso term is where prior knowledge is included in the model. Two parameters affect the lasso term: α and λ (see Equation (3.19)). Here, α is the weight given to the

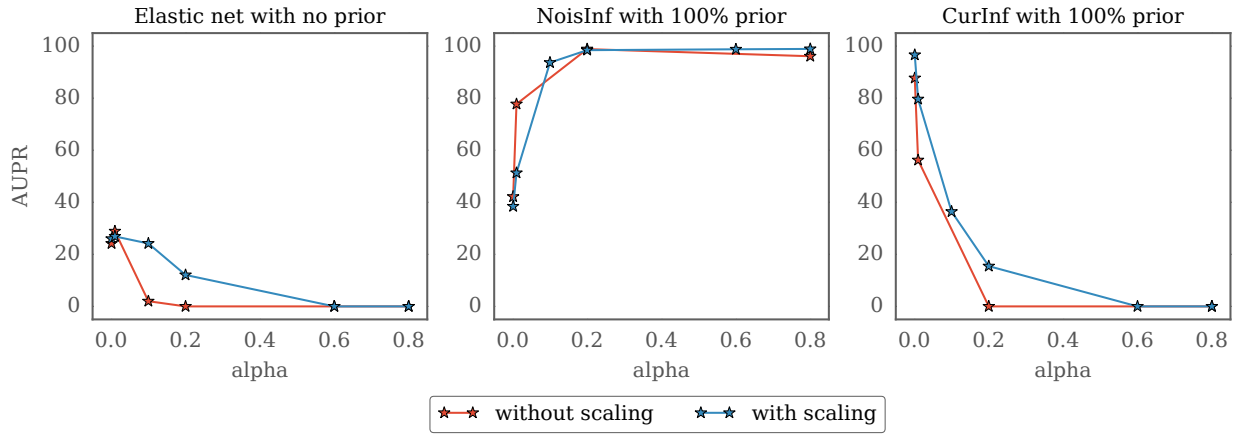


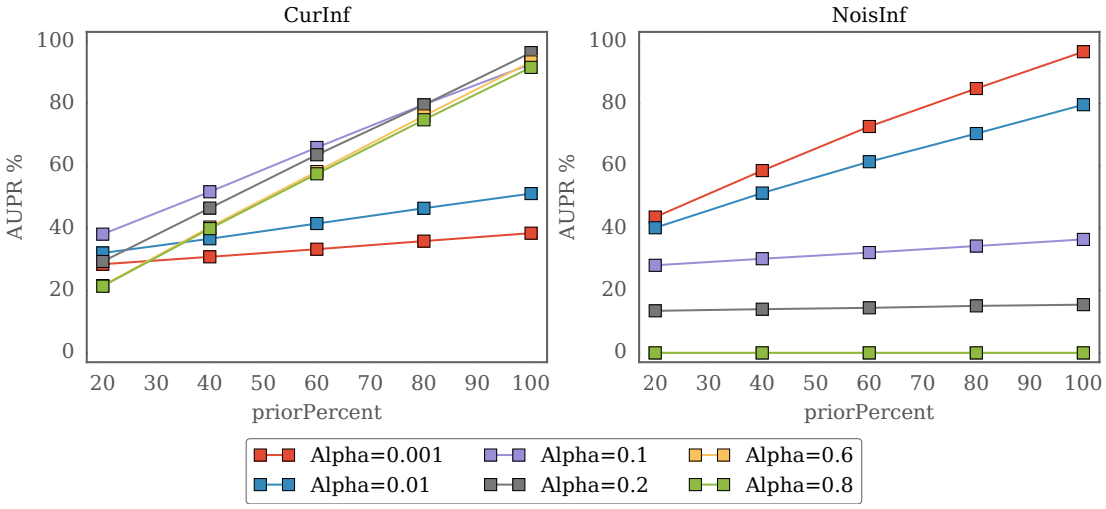
Figure 6.6: Effect of scaling the features on the choice of the regularization parameters α in the in silico data set. The first graph is the elastic net without prior knowledge, the second is penalty scaling with 100% prior knowledge and the third is feature scaling with 100% prior knowledge.

penalty term in general, and λ is the balancing parameters between the lasso term and the ridge term. We found that the choice of α has more effect on the accuracy than the choice of the lasso term ratio. The lasso term λ is fixed to be 0.5 throughout the rest of the analysis. The accuracy is sensitive to the choice of α as shown in Figure 6.7.

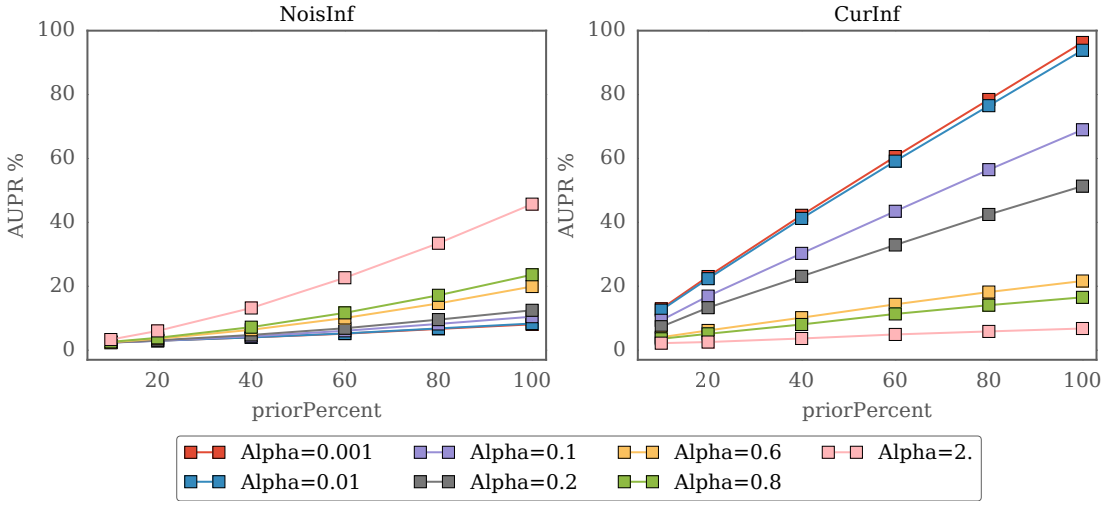
Cross-validation for the choice of α without any bounds can produce bad accuracy. Choosing α for our proposed method CurInf is easier than NoisInf proposed by [31] as shown in Figure 6.7 because prior knowledge is added in the main error term, and small values of α (typically less than 0.1) give more focus on the error term, with low weight on the penalty terms.

6.7 Using Bounded Cross-validation

We tested our proposed heuristic to provide a bounded search space for the weight of the penalty term. Both CurInf and NoisInf uses cross-validation (CV) to find the penalty term weight parameter α in the elastic net, Equations (3.19) and (3.20), when incorporating



(a) In silico (DREAM5)



(b) *E. coli*

Figure 6.7: Effect of varying penalty term weight α on the accuracy in NoisInf and CurInf inference algorithm (with lasso term weight λ is fixed).

prior knowledge. Figure 6.8 shows the accuracy using bounded CV versus unrestricted CV for both CurInf and NoisInf. Applying bounded CV improved the accuracy for CurInf when the prior knowledge is not well supported by the gene expression data as in the *E. coli* and *S. cerevisiae* data sets.

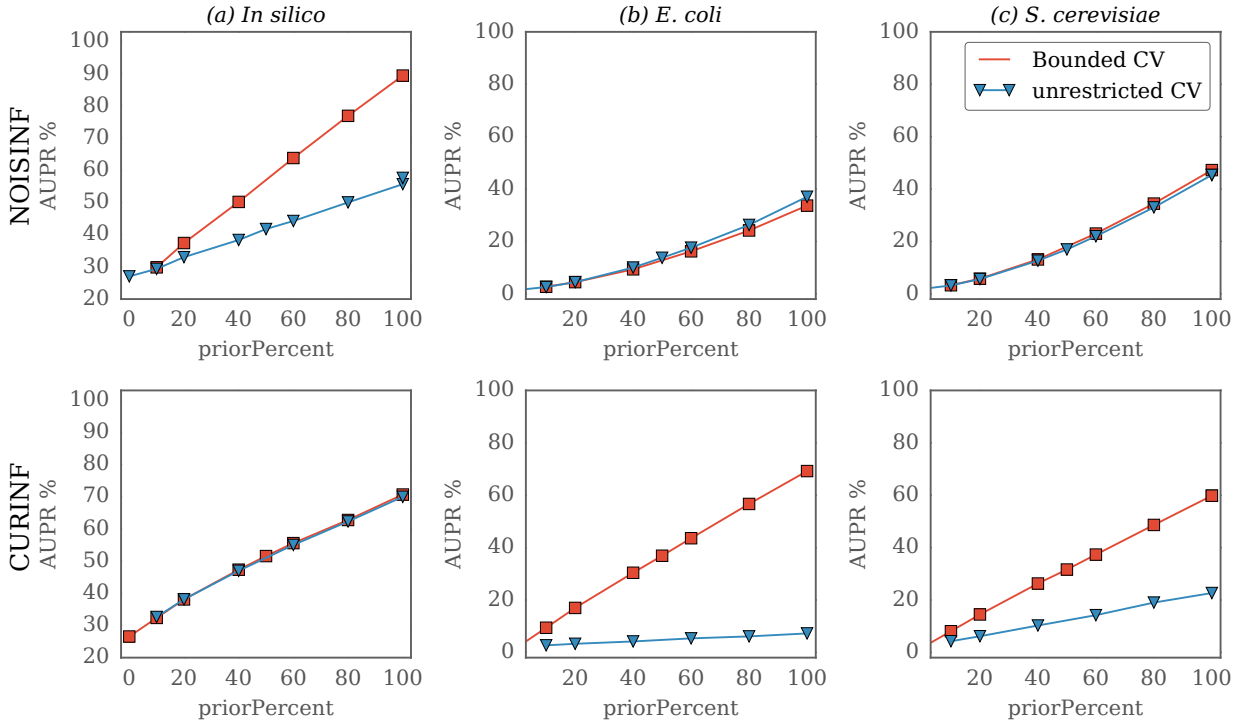


Figure 6.8: Using bounded versus unrestricted cross-validation for finding the parameter α in NoisInf and CurInf.

6.8 Heuristic to Improve Execution Time

The heuristic described in [51] is tested in feature scaling to improve execution time. In this heuristic, each gene is limited to have the top 30 transcription factors with the highest mixed-CLR values as potential transcription factors in the fine-grain prediction of the Elastic Net instead of using all transcription factors. Testing is performed using an Intel Core i7, 2.2GHz processor with 4 cores. This filtering improves running time considerably as shown

in Table 6.2. The gain in the accuracy of using all transcription factors compared to using the top 30 transcription factors is on average 1%.

Data set	Using all predictors	Using top 30 predictors
In silico	20	4
E. coli	840	15

Table 6.2: Average running time in minutes

Chapter 7

Case Studies and User Feedback

In this chapter, we present feedback collected from potential users of DeTangle who are researcher in the fields of computer science, biology, and/or bioinformatics. We also introduce a case study in *Arabidopsis thaliana* root stem cells obtained from collaboration with a biologist in the Department of Crop and Soil Environmental Sciences at Virginia Tech using DeTangle. Afterwards, we show another case study in *Escherichia coli* in which we surveyed the literature to validate some of the regulatory relations predicted with high confidence by PEAK.

7.1 DeTangle User Feedback

We collected user feedback from 10 users who resemble the likely users of DeTangle. The users are researchers with a biology major, with a biological sciences major, from the Genetics, Bioinformatics and Computational Biology (GBCB) program, from the Department of Crop and Soil Environmental Sciences, from the Department of Computer Science with research in computational biology and bioinformatics, and from the Department of Computer

Science with research in visualization. The users are from three universities: Virginia Tech, University of Washington, and Alexandria University in Egypt.

7.1.1 Procedure

A meeting was scheduled with each user individually in Torgerson Hall at the Computational Biology and Bioinformatics lab. I spent from 40 to 60 minutes with each user. The same procedure was performed with all except three users. The first exception is the professor from the Department of Crop and Soil Environmental Sciences who was given the link of the server and had an extended amount of time to use it with his own expression data from *Arabidopsis thaliana* (which we use as the default demo in our system). Consequently, I had an extended meeting to discuss the results and the findings based on his data. This special case study is presented in Section 7.2. Two users who are not from Virginia Tech were given the link to the server and their feedback was collected.

First, the purpose of the DeTangle prediction server was explained to the user at a high level without explaining how to use the interface. The purpose of not providing the user with any initial demo is to find which features are not initiative or easy to use. Next, the user opened the system on their laptops (except two users used my machine). After that, the user was asked to explore and use the system freely. I took notes, observations, suggestions, and feature requests while the user examined the system. Eventually, some guidance was given to the users to examine and test unexplored features.

7.1.2 Requested Features

The following are existing features that were noted by most users as useful or interesting. Those features are explained in detail in Chapter 5.

- The overall system is “intuitive” and “neat”.
- The ability to add feedback or to judge predictions (that we call post-prediction prior knowledge).
- The server re-evaluates the network quickly (i.e., runs in realtime).
- Providing a ready demo example.
- Filtering edges using the confidence slide bar.
- Highlighting and showing gene neighbors.
- Showing and hiding genes to explore the network and reduce its size.
- A link to GeneBank for more information about selected genes.
- Showing gene expression with the GRN.
- Providing right-click for quick access to features (i.e., context menus).

We have implemented most of the requested users’ features and enhancements, some of which are explained bellow.

- Include a button to reset all added or deleted edges by the user.
- Provide a demo video to show all possible features of the system.
- “Save As”: Ability to save current network as a new network on the server.
- Clear colored edges of the previous online-PEAK evaluation.
- Use different colors for updated online-PEAK edges and mouse-selected edges.
- Some wording changes and locations of some features in the main menu.

Finally, some features were requested by only one user, had contradicting opinions between users, or need extended amount of time to design and implement. The following features are thus suggested for future work.

- Include a fixed legend on the screen explaining edge colors (in addition to the existing help).
- Double-click to highlight nodes.
- Export the graph in a publication quality format such as EPS or SVG.
- Box selection for selecting multiple genes (while others preferred the current dragging of the whole canvas). We will include both desired features by allowing a toggle button between the two features, since the visualization library used, Cytoscape.js, does not support both at the same time.
- Investigate the location of the link to the sample demo. A few users were not able to find it in the first 30 seconds.

One of the interesting requested features is to provide a way to compare the network before and after online-PEAK evaluation. One proposed solution is to include a snapshot of the network before each evaluation and display it in an extra tab. Another solution is to investigate using multiple color coding for returned edges.

7.2 Case Study in *A. thaliana* Root Stem Cells

In this section, I present a case study in collaboration with Dr. Song Li at Virginia Tech that illustrates a useful use-case of the DeTangle system to predict gene regulatory network using PEAK. The data set used for this case study is the *Arabidopsis thaliana* root stem cell

niche (RSCN) gene regulatory network. The RSCN has a well studied regulatory network and is regarded as a model system for development and stem cell analysis in plants [67]. This is due to being simple and easily available for modular experiments.

The subnetwork shown in Figure 7.1 was originally presented by Azpeitia et al. [4]. In their paper, they attempt to predict the missing interactions in the experimentally verified *Arabidopsis thaliana* RSCN regulatory network using a Boolean network approach and subsequently validate their predicted network dynamics with experimental data. We used their validated RSCN gene regulatory network as the gold standard for this case study.

As the RSCN study [4] did not provide their gene expression data data, we used other available expression data. We used 21 gene expression experiments provided by Professor Song Li in Crop and Soil Environmental Sciences at Virginia Tech. The data was obtained using cell type specific as well as developmental zone specific gene expression data generated with RNA-Seq experiments from *Arabidopsis thaliana* roots [47]. No data was available for mR165 or AUXIN, so we excluded them from this analysis.

The regulatory network in Figure 7.1 includes 9 genes and 26 regulatory relations (activators and inhibitors, including self-regulators). The figure is extracted from the DeTangle Web interface. DeTangle's parallel plot of the gene expression data of the 9 genes is shown in Figure 7.2.

DeTangle was used to predict the gene regulatory network for the *Arabidopsis thaliana* root stem cell genes, and the resulting network is shown in Figure 7.3. Only edges with confidence greater than or equal to 0.05 are shown. True positive edges are colored in bold green. There were 22 predicted edges and the AUPR accuracy of this prediction is 64%. No prior knowledge was used to predict this network. For example, DeTangle was able to correctly uncover all regulators of PHB, SHR, SCR, and MGP. On the other hand, DeTangle could not predict the regulators of WOX5 correctly. We think the reason is that WOX5 was

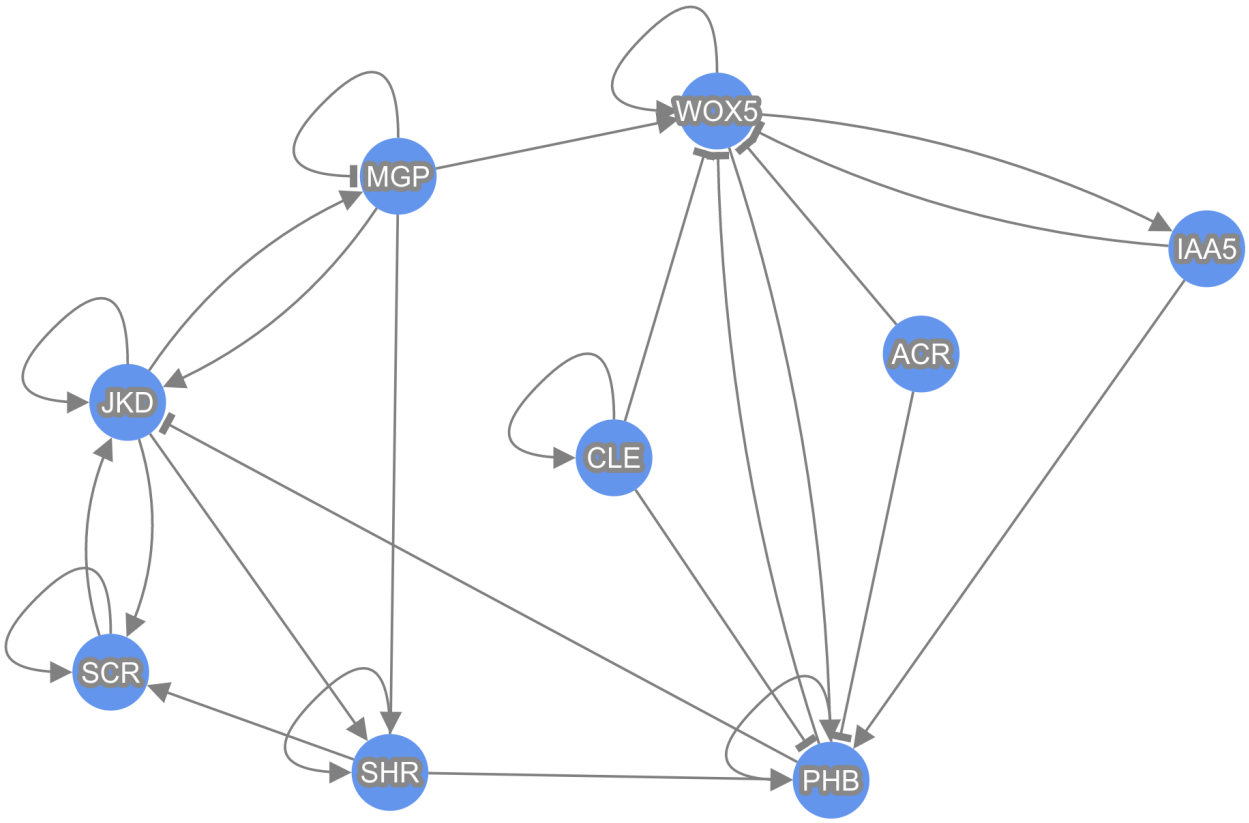


Figure 7.1: This figure shows the known gene regulatory network (gold standard) in *Arabidopsis thaliana* root stem cell niche (RSCN), extracted from Azpeitia et al. [4]. The subnetwork includes 9 genes and 26 regulatory relations.

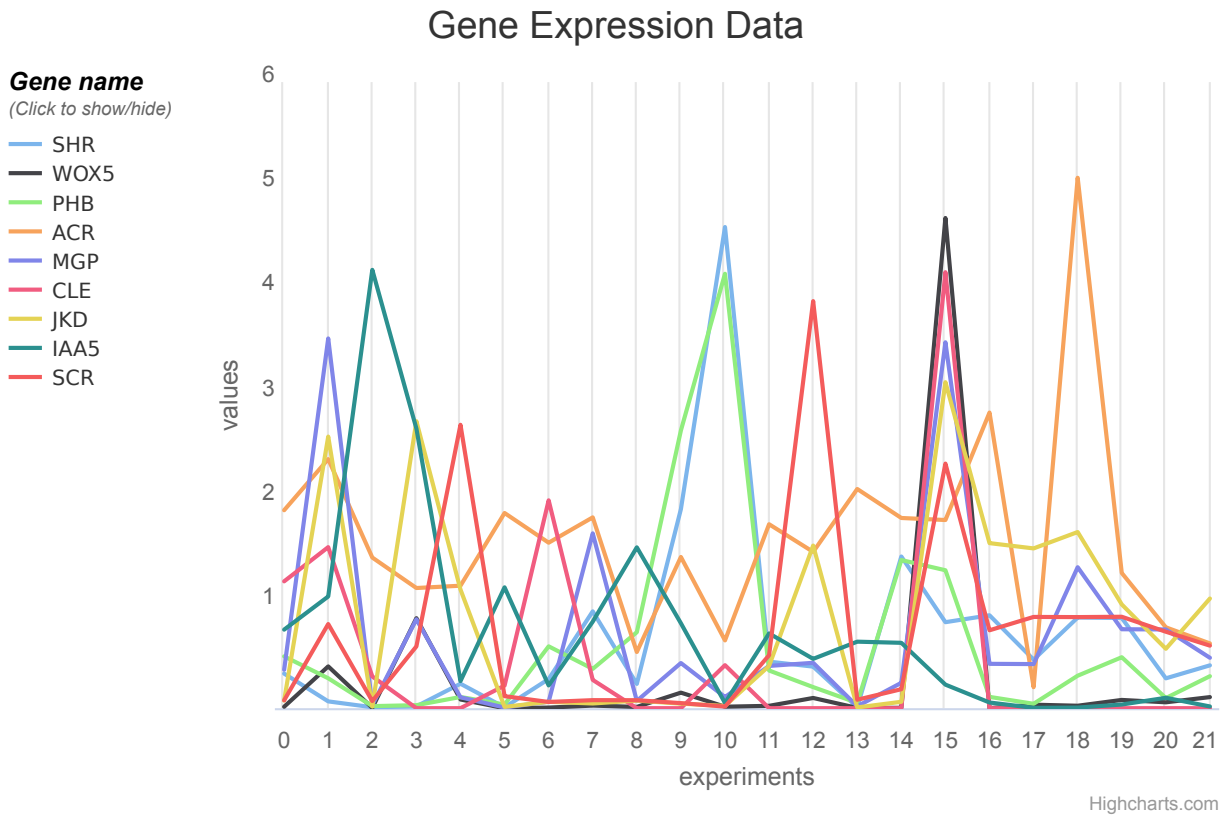


Figure 7.2: A parallel plot of the gene expression data for 21 experiments from Li et al. [47] as presented by DeTangle.

not expressed in most of the expression data that we used in this study, which can be seen in the expression data of WOX5 in Figure 7.6. Note that our ODE method does not predict self regulations as shown in the equations in Section 3.3.

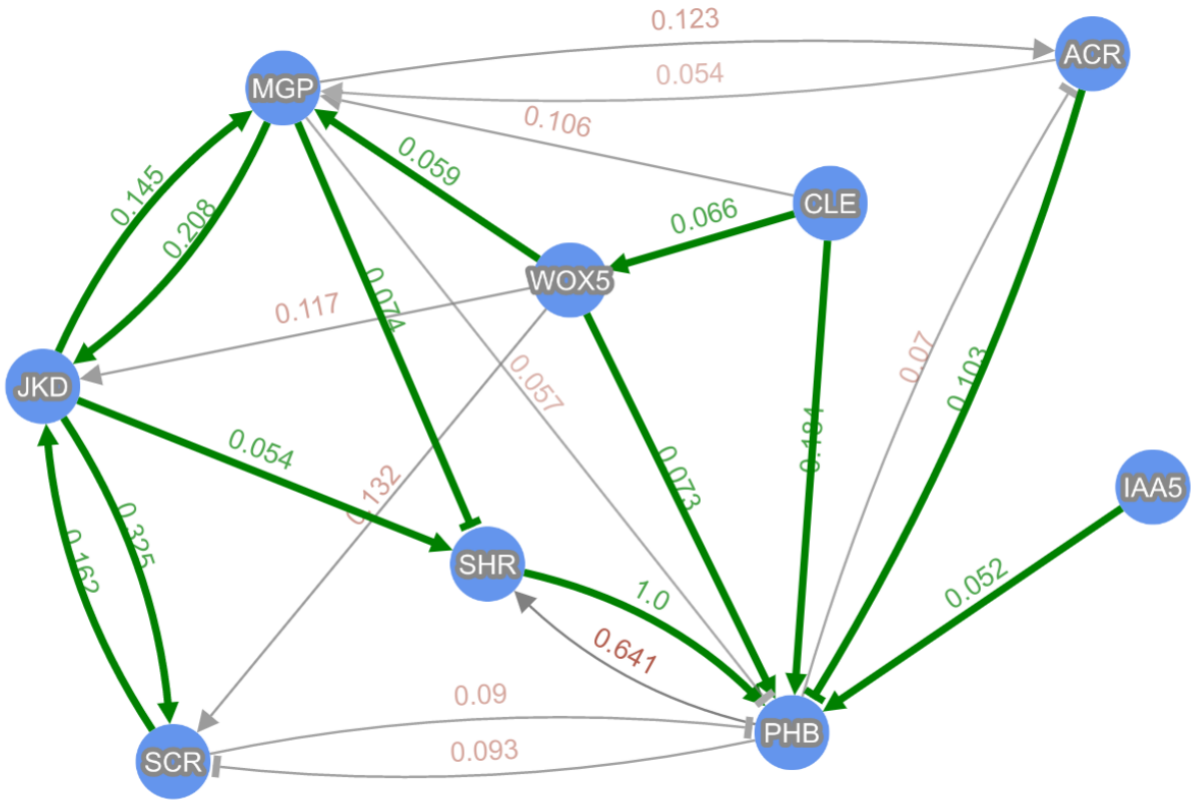


Figure 7.3: The predicted regulatory relations for the *Arabidopsis thaliana* root stem cell niche using our system DeTangle. Green edges represent the true positive edges, i.e., relations that exist in both the gold standard network and the predicted network.

In addition to the true positive network, DeTangle predicted additional relations with high confidence. The same predicted network in Figure 7.3 is shown again in Figure 7.4 but using a higher edge threshold of 0.1, i.e., only edges with confidence greater than or equal to 0.1 are shown. The edges colored in bold green are relations that have high prediction confidence but do not exist in the gold standard network (assumed false positives compared to the gold standard). For example, CLE is predicted to activate MGP. The parallel plot of gene expression data shown in Figure 7.5 supports this hypothesis. Also, WOX5 is predicted

to regulate JKD and SCR; this observation is also supported by their gene expression data in Figure 7.6. Those predictions with high confidence can be used as potential directions for further experimental studies to the *Arabidopsis thaliana* root stem cell system. Dr. Song Li's research lab is considering those newly predicted regulatory interactions for further study and analysis.

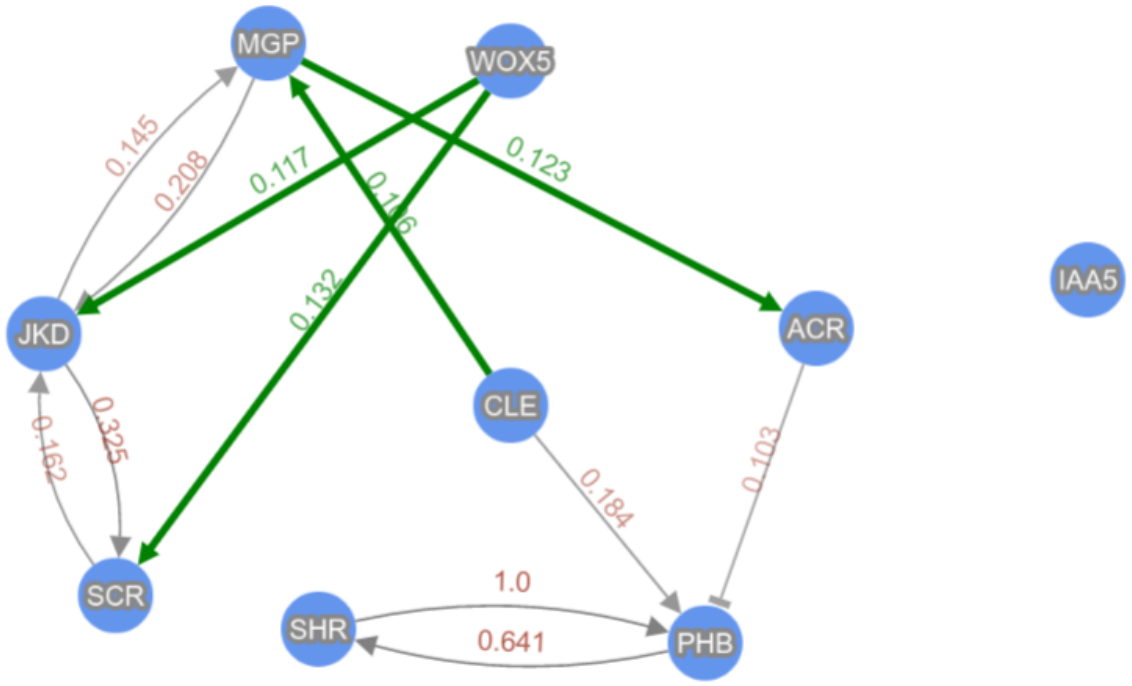


Figure 7.4: High confidence predicted regulatory relations (> 0.1) in the *Arabidopsis thaliana* root stem cell niche. New predicted edges that are not in the gold standard network are shown in bold-green.

7.3 Case Study in *E. coli*

I studied the ability of DeTangle to discover new regulatory relations that are not part of the gold standard. Those regulatory relations can be used as potential subjects for experimental

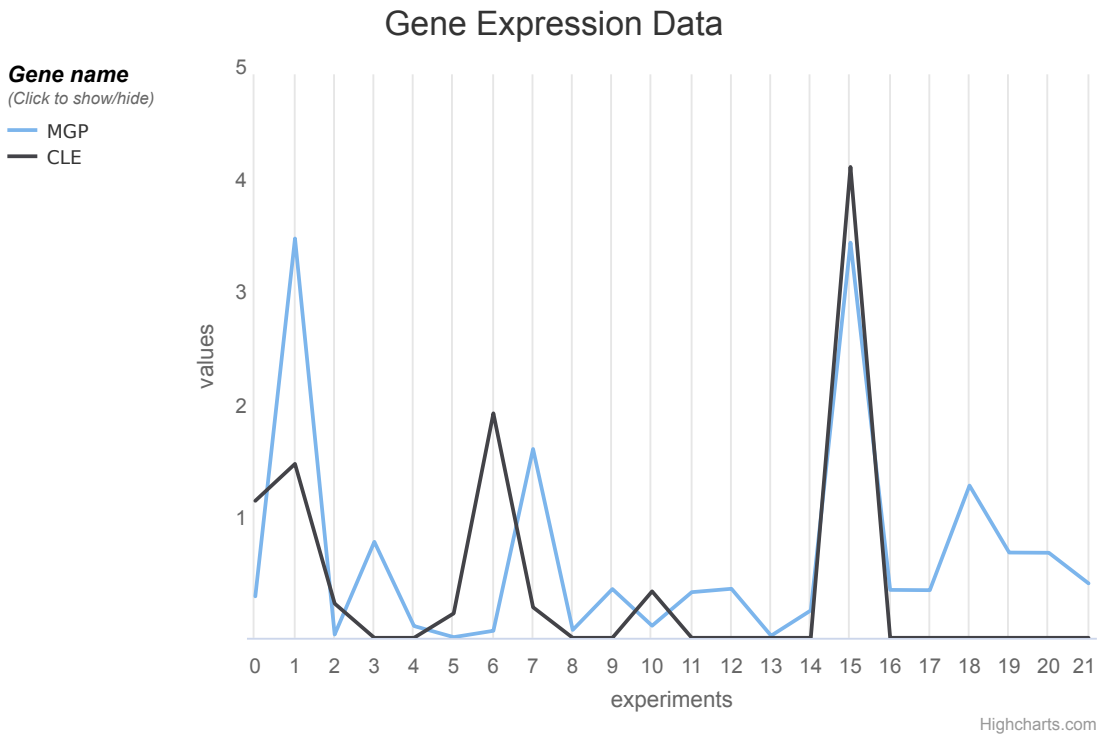


Figure 7.5: Gene Expression shows support for the predicted regulation of CLE which activates MGP in the *Arabidopsis thaliana* root stem cell niche.

Figure 7.6: Gene Expression shows support for the predicted regulatory relation of WOX5 activating JKD and SCR in the *Arabidopsis thaliana* root stem cell niche. Experiments 1 through 5 shows a weaker support.

testing by biologists. I used a case study from the *E. coli* data set to manually inspect the resulting gene regulatory network.

Adding prior knowledge enabled PEAK to discover new validated interactions that are not in the gold standard. Since the gold standard for the *E. coli* benchmark was created in 2010 from RegulonDB version 7, new interaction have been added to the database. We validated CurInf predictions that are considered false positives according to the gold standard using the most recent RegulonDB version 9. We were able to validate 128 of the computationally inferred interactions, including 17 with strong experimental evidence. For example, in the *E. coli* predicted GRN using 100% of the gold standard as prior knowledge, CurInf predicted Lrp as a TF for *cadA*, which is not in the gold standard. We validated this regulatory relation, and it was recently published that Lrp is a regulator of *cadA* [66]. Similarly, CurInf predicted MarA to be a TF for *ybjC*. We found that this regulatory relation was validated in a study by Martin et al. [54].

Chapter 8

Future Work

8.1 Ensemble of Several Methods in DeTangle

As mentioned in Section 4.2.4, other methods can be used in online-PEAK if they satisfy the described requirements. We found that the recent prediction algorithm BGRMI [38] satisfies online-PEAK requirements and thus can be integrated with the DeTangle system. Furthermore, it is possible to develop an ensemble between the predictions of PEAK and BGRMI to produce more reliable GRN inference. Ensembles of several GRN inference methods was previously shown to improve regulatory network prediction accuracy [52].

8.2 Integration with Simulation Algorithms

Biological networks simulators are an important tool in systems biology to construct models and hypotheses on pathways and inspect their effect [19]. As a future work, it is possible to integrate simulation software with gene regulatory network inference. Selected regulatory relations that are predicted using DeTangle can be tested in a simulation tool to observe

their effect on pathways of interest.

8.3 Visualization of Large Scale GRN

In recent years, the field of visual analytics has had many advances in the analysis of complex networks such as social networks and epidemiology. Moreover, several libraries have become available for fast development of visualizations and for the analysis of graphs. We propose a system for the visualization and exploration of large scale gene networks to enable reaching interesting parts of the network using filtering and zooming.

8.4 Proposed Dynamic Query Visualization System

As a future work, DeTangle will be an interactive visualization and exploration framework for complex protein networks such as protein-protein interactions and gene regulatory networks. The output of the inference engine in the first part of this thesis can be visualized in an easy to explore format using a dynamic query visualization engine. The dynamic query engine uses several computational and analytical algorithms to produce the interactive visualization. The main visualization strategy to be used is the "overview and expand on demand" concept [59]. In this approach, the entire network is not displayed, but only an "overview" of interesting subgraphs. Those subgraphs are inferred by mining the large networking using clustering and graph partitioning. Afterwards, subgraphs can be selected for expansion and further exploration (more focused view). In the overview, only cluster centers and hubs are shown to the user when the total graph size (number of nodes) is larger than a threshold. The user can select a cluster to expand. Furthermore, genes within each cluster can be queried and filtered using several available criteria to reach a smaller subgraph. We call this filtering feature dynamic queries.

In our proposed visualization engine, the overview is created using network alignment and clustering as follows. The regulatory GRN is aligned with a well known network of a selected model organism. The user can choose one of several predefined model organisms in the system. Next, the network is partitioned into possibly overlapping subgraphs based on the alignment and the clustering results. Clustering of large networks involves the detection of overlapping communities [62]. Figure 8.1 shows the components and the flow of the proposed system.

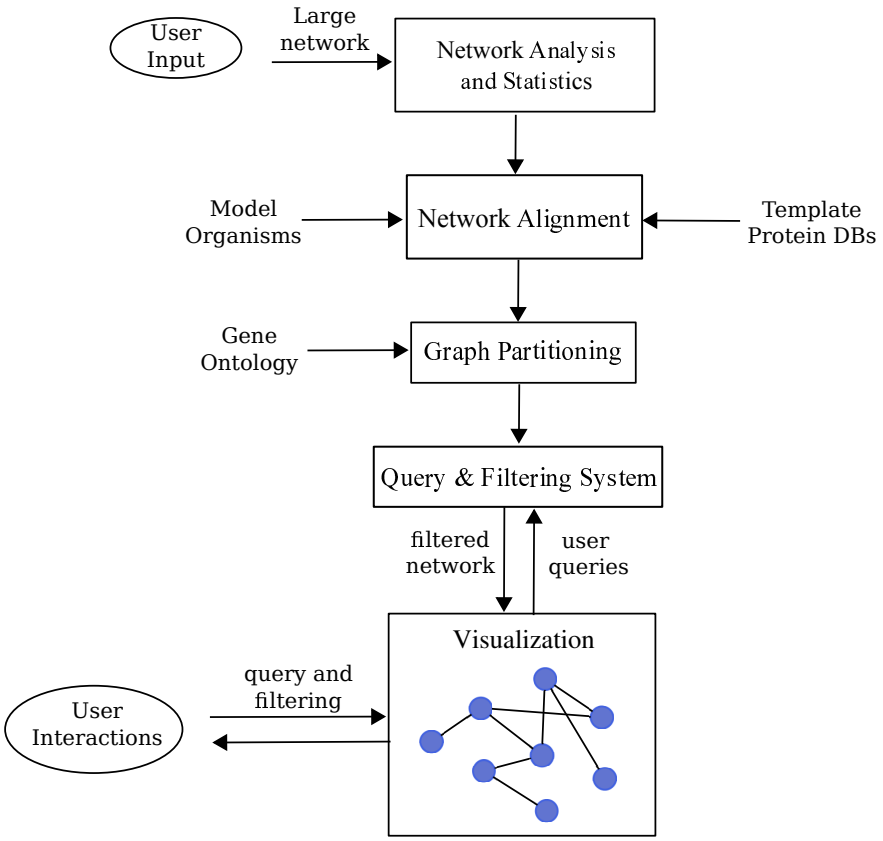


Figure 8.1: The proposed interactive visualization system for complex gene regulatory networks.

The focused subgraphs that are reached after dynamic queries are editable. The expert user can modify the graph by several user interactions, such as adding and removing edges, and also changing weights of edges.

Chapter 9

Conclusion

In this work, I developed PEAK, a system to integrate both noisy and curated structure prior knowledge in the same GRN inference model based on the Inferelator algorithm [9]. Curated or reliable prior knowledge comes from experimentally verified pathways, whereas noisy prior knowledge can be any supporting information about the network structure that does not necessarily imply regulatory relations, such as PPI. To my knowledge, the integration of curated prior knowledge has not been addressed by any previous method, especially when the prior knowledge is not well supported by the gene expression data. I have shown that such poor prior knowledge support exists in real data compared to synthetic data. I present a novel method in GRN inference, CurInf, to integrate curated prior knowledge from experimentally validated TF-target interactions. For noisy prior knowledge, I use a similar approach to the robust method MEN [31], which I call NoisInf. Both CurInf and NoisInf add prior knowledge to the model but in different ways that account for the confidence given to the prior knowledge.

My system DeTangle enables the domain expert to interact with my online-PEAK inference algorithm through a visual interface to add their knowledge as post-prediction feedback. The

predicted GRN is displayed to the biologist in a Web interface, with the ability to modify edges and submit changes back to the prediction algorithm for re-evaluation. DeTangle uses the inference method PEAK as the underlying online algorithm.

My testing on synthetic data as well as real data from *E. coli* and *S. cerevisiae* shows that adding incremental prior knowledge significantly improved the accuracy of the predicted GRN of PEAK and online-PEAK. The inference algorithm is robust to noisy interactions, and noisy prior knowledge will not affect the inference accuracy if they are not supported by the gene expression data. I obtained useful user feedback from potential users of DeTangle system. I have also presented a case-study in *Arabidopsis thanlian* root stem cells, with 4 DeTangle predicted interactions chosen for further study by the biologist.

Moreover, CurInf is superior to NoisInf in the integration of curated prior knowledge, even if the prior knowledge is not well supported by the gene expression data. The improvement is more pronounced in the *E. coli* and *S. cerevisiae* data compared to the synthetic data, since my method is designed to address gene expression data that may poorly support the prior knowledge. The difference in accuracy between synthetic and real data can be due to the expected complexity of the latter and the fact that the synthetic expression data was generated from a known network while the regulatory networks of the *E. coli* and the *S. cerevisiae* are still not complete. It is possible that true TF-target interactions exist among the predicted interactions that are considered false positives, which reduces the AUPR score.

Thus, better integration of prior knowledge improves the accuracy and enables the inference algorithm to predict new potential regulatory relations. High ranked predicted interactions, especially with a consensus of multiple computational methods, can be potential subjects for biologists to test and validate experimentally. GRN can help in studying complex molecular interactions, understanding diseases, and aiding drug design.

Appendices

Appendix A

Input Data Format

A.1 Gene Expression

The expression data file contains the matrix of gene expression values. Each row corresponds to a an experiment sample, and each column to a gene. In other words, element (i, j) is the expression value of gene j in experiment i. The first line of the file gives the unique gene name or label for every column. The expression data should be already normalized with the user's preferred normalization method.

A.2 Experiments Metadata

The metadata file contains meta information for each gene expression experiment. The information is presented as a matrix, where rows correspond to gene expression samples and columns to the description their features. Row k gives the features for the corresponding row k in the uploaded expression data file. There are four columns for the following optional features:

Line No.	Experiment	DeletedGenes	Time	Repeat
1	1	NA	NA	1
2	1	NA	NA	2
3	2	NA	NA	1
4	2	NA	NA	1
5	2	NA	NA	1
6	3	NA	0	1
7	3	NA	30	1
8	3	NA	60	1
9	3	G5	30	1
10	3	G5	60	1
11	4	G5,G8	NA	1
12	5	NA	NA	1
13	5	NA	NA	1

Table A.1: Example metadata file.

- Experiment: an identifier for the experiment (can be a number).
- DeletedGenes: a list of deleted (knocked out) genes.
- Time: the time point for experiments that are part of a time series (use "NA" if not time series).
- Repeat: used to distinguish experimental replicates.

The value NA can be used when any feature is not applicable or not available.

Table **??** illustrates and example metadata file with thirteen samples, which were obtained from five different experiments. Samples that are from the same experiment were done in the exact experimental setting.

The first column in Table **??** shows the line numbers and is not part of the actual metadata file. Sample 1 and 2 (first and second row) are replicates of experiment 1, as indicated by the repeat numbers in the right most column. Samples 3, 4, and 5 were all obtained using the same experimental setting (experiment 2). Samples 6-10 are part of a time-series (experiment

3). Sample 6 is the initial time point at $t = 0$. Samples 9 and 10 show the expression profile at time $t = 30$ and $t = 60$ after knockout of gene G5. Samples 7 and 8 are the corresponding controls without knockout of G5. If two genes are perturbed simultaneously, as in the double knockout of genes G5 and G8 sample 11 (experiment 4), they are listed separated by a comma. Note that experiments that are not part of a time series (time is "NA") are often done at a time when the perturbation is assumed to have unfolded its full effect (i.e., in a steady state).

A.3 Transcription Factors

The transcription factors (TFs) file list the genes of the network that are potential TFs. Only genes that are part of this list will be included as regulators in the predicted network. Naturally, a TF may regulate another TF gene (TFs can also be targets in the network). This list should include all potential TFs since genes not included in the list will not be regarded by the algorithm as TFs. On the other hand, providing a TFs file can greatly improve the prediction. This file is optional; if not provided, all genes will be considered potential TFs (although this is not recommended).

A.4 Prior Knowledge

The prior knowledge file is optional and includes known genes and their TFs. The first column is the TF and the second is the corresponding target gene. The third column is 1 if the relation is activation, and -1 if the relation is inhibition. The file has no headers.

Bibliography

[1] Doaa Altarawy, Fatma-Elzahraa Eid, and Lenwood S. Heath. PEAK: Integrating curated and noisy prior knowledge in gene regulatory network inference. *Journal of Computational Biology*, 24, 2017.

[2] Gökmen Altay and Frank Emmert-Streib. Inferring the conservative causal core of gene regulatory networks. *BMC Systems Biology*, 4(1):132, 2010.

[3] Emmanouil Athanasiadis, Marilena Bourdakou, and George Spyrou. D-Map: Random walking on gene network inference maps towards differential avenue discovery. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 14(2):484–490, 2016.

[4] Eugenio Azpeitia, Nathan Weinstein, Mariana Benítez, Luis Mendoza, and Elena R. Alvarez-Buylla. Finding missing interactions of the *Arabidopsis thaliana* root stem cell niche gene regulatory network. *Frontiers in Plant Science*, 4:110, 2013.

[5] Ozgun Babur, Ugur Dogrusoz, Emek Demir, and Chris Sander. ChiBE: Interactive visualization and manipulation of BioPAX pathway models. *Bioinformatics*, 26(3):429–431, 2010.

[6] Tanya Barrett, Stephen E. Wilhite, Pierre Ledoux, Carlos Evangelista, Irene F. Kim, Maxim Tomashevsky, Kimberly A. Marshall, Katherine H. Phillippy, Patti M. Sherman,

Michelle Holko, et al. NCBI GEO: archive for functional genomics data sets—update. *Nucleic Acids Research*, 41(D1):D991–D995, 2013.

[7] Aaron Barsky, Tamara Munzner, Jennifer Gardy, and Robert Kincaid. Cerebral: Visualizing multiple experimental conditions on a graph with biological context. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1253–1260, 2008.

[8] Benjamin M. Bolstad, Rafael A. Irizarry, Magnus Åstrand, and Terence P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–193, 2003.

[9] Richard Bonneau, David J. Reiss, Paul Shannon, Marc Facciotti, Leroy Hood, Nitin S. Baliga, and Vesteinn Thorsson. The Inferelator: An algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo. *Genome Biology*, 7(5):R36, 2006.

[10] Bobby-Joe Breitkreutz, Chris Stark, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, Michael Livstone, Rose Oughtred, Daniel H Lackner, Jürg Bähler, Valerie Wood, et al. The BioGRID interaction database: 2008 update. *Nucleic Acids Research*, 36(suppl 1):D637–D640, 2008.

[11] Kevin R. Brown and Igor Jurisica. Unequal evolutionary conservation of human protein interactions in interologous networks. *Genome Biology*, 8(5):R95, 2007.

[12] Kevin R. Brown, David Otasek, Muhammad Ali, Michael J. McGuffin, Wing Xie, Baiju Devani, Ian Lawson van Toch, and Igor Jurisica. NAViGaTOR: Network analysis, visualization and graphing Toronto. *Bioinformatics*, 25(24):3327–3329, 2009.

[13] Atul J. Butte, Isaac S. Kohane, and I. S. Kohane. Mutual information relevance networks: Functional genomic clustering using pairwise entropy measurements. In *Pacific Symposium on Biocomputing*, volume 5, pp. 418–429, 2000.

[14] Ethan G. Cerami, Benjamin E. Gross, Emek Demir, Igor Rodchenkov, Özgün Babur, Nadia Anwar, Nikolaus Schultz, Gary D. Bader, and Chris Sander. Pathway Commons, a Web resource for biological pathway data. *Nucleic Acids Research*, 39(suppl 1):D685–D690, 2011.

[15] Guocai Chen, Michael J Cairelli, Halil Kilicoglu, Dongwook Shin, and Thomas C Rindflesch. Augmenting microarray data with literature-based knowledge to enhance gene regulatory network inference. *PLoS Comput Biol*, 10(6):e1003666, 2014.

[16] Kam D. Dahlquist, John David N. Dionisio, Ben G. Fitzpatrick, Nicole A. Anguiano, Anindita Varshneya, Britain J. Southwick, and Mihir Samdarshi. GRNsight: A Web application and service for visualizing models of small-to medium-scale gene regulatory networks. *PeerJ Computer Science*, 2:e85, 2016.

[17] Sayoni Das, Ian Sillitoe, David Lee, Jonathan G. Lees, Natalie L. Dawson, John Ward, and Christine A. Orengo. CATH FunFHMMer Web server: Protein functional annotations using functional family assignments. *Nucleic Acids Research*, 43(W1):W148–W153, 2015.

[18] Carsten O. Daub, Ralf Steuer, Joachim Selbig, and Sebastian Kloska. Estimating mutual information using B-spline functions– An improved similarity measure for analysing gene expression data. *BMC Bioinformatics*, 5(1):118, 2004.

[19] Hidde De Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of computational biology*, 9(1):67–103, 2002.

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

[21] Alptug Dilek, Mehmet E. Belviranli, and Ugur Dogrusoz. VISIBIOweb: Visualization

and layout services for BioPAX pathway models. *Nucleic Acids Research*, p. gkq352, 2010.

[22] Ugur Dogrusoz, Mehmet E. Belviranli, and Alptug Dilek. CiSE: A circular spring embedder layout algorithm. *IEEE Transactions on Visualization and Computer Graphics*, 19(6):953–966, 2013.

[23] 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.

[24] Max Franz, Christian T. Lopes, Gerardo Huck, Yue Dong, Onur Sumer, and Gary D. Bader. Cytoscape.js: A graph theory library for visualisation and analysis. *Bioinformatics*, 32(2):309–311, 2015.

[25] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1, 2010.

[26] Socorro Gama-Castro, Heladia Salgado, Alberto Santos-Zavaleta, Daniela Ledezma-Tejeida, Luis Muñiz-Rascado, Jair Santiago García-Sotelo, Kevin Alquicira-Hernández, Irma Martínez-Flores, Lucia Pannier, Jaime Abraham Castro-Mondragón, et al. RegulonDB version 9.0: High-level integration of gene regulation, coexpression, motif clustering and beyond. 44(D1):D133–D143, 2016.

[27] Nils Gehlenborg, Seán I. O'Donoghue, Nitin S. Baliga, Alexander Goesmann, Matthew A. Hibbs, Hiroaki Kitano, Oliver Kohlbacher, Heiko Neuweger, Reinhard Schneider, Dan Tenenbaum, and Others. Visualization of omics data for systems biology. *Nature Methods*, 7:S56–S68, 2010.

[28] Begum Genc and Ugur Dogrusoz. An algorithm for automated layout of process description maps drawn in SBGN. *Bioinformatics*, 32(1):77–84, 2016.

[29] Burkay Genç and U Dogrusoz. A layout algorithm for signaling pathways. *Information Sciences*, 176(2):135–149, 2006.

[30] Burkay Genc and Ugur Dogrusoz. A constrained, force-directed layout algorithm for biological pathways. In *International Symposium on Graph Drawing*, pp. 314–319. Springer, 2003.

[31] Alex Greenfield, Christoph Hafemeister, and Richard Bonneau. Robust data-driven incorporation of prior knowledge into the inference of dynamic regulatory networks. *Bioinformatics*, 29(8):1060–1067, 2013.

[32] Alex Greenfield, Aviv Madar, Harry Ostrer, and Richard Bonneau. DREAM4: Combining genetic and dynamic information to identify biological networks and dynamical models. *PloS One*, 5(10):e13397, 2010.

[33] Anne-Claire Haury, Fantine Mordelet, Paola Vera-Licona, and Jean-Philippe Vert. TIGRESS: Trustful inference of gene regulation using stability selection. *BMC Systems Biology*, 6(1):145, 2012.

[34] Feng He, Rudi Balling, and An-Ping Zeng. Reverse engineering and verification of gene networks: Principles, assumptions, and limitations of present methods and future perspectives. *Journal of Biotechnology*, 144(3):190–203, 2009.

[35] Michael Hecker, Sandro Lambeck, Susanne Toepfer, Eugene Van Someren, and Reinhard Guthke. Gene regulatory network inference: Data integration in dynamic models—a review. *Biosystems*, 96(1):86–103, 2009.

[36] Ivan Herman and M. Scott Marshall. GraphXML— An XML-based graph description format. In *International Symposium on Graph Drawing*, pp. 52–62. Springer, 2000.

[37] Graham J. Hickman and T. Charlie Hodgman. Inference of gene regulatory networks using boolean-network inference methods. *Journal of Bioinformatics and Computational Biology*, 7(06):1013–1029, 2009.

[38] Luis F. Iglesias-Martinez, Walter Kolch, and Tapesh Santra. BGRMI: A method for inferring gene regulatory networks from time-course gene expression data and its application in breast cancer research. *Scientific Reports*, 6, 2016.

[39] Rafael A. Irizarry, Benjamin M. Bolstad, Francois Collin, Leslie M. Cope, Bridget Hobbs, and Terence P. Speed. Summaries of Affymetrix GeneChip probe level data. *Nucleic Acids Research*, 31(4):e15–e15, 2003.

[40] Alexandre Irrthum, Louis Wehenkel, and Pierre Geurts. Inferring regulatory networks from expression data using tree-based methods. *PloS One*, 5(9):e12776, 2010.

[41] Senol Isci, Haluk Dogan, Cengizhan Ozturk, and Hasan H. Otu. Bayesian network prior: Network analysis of biological data using external knowledge. *Bioinformatics*, 30(6):860–867, 2014.

[42] Yeongjun Jang, Namhee Yu, Jihae Seo, Sun Kim, and Sanghyuk Lee. MONGKIE: An integrated tool for network analysis and visualization for multi-omics data. *Biology Direct*, 11(1):10, 2016.

[43] Björn H. Junker, Christian Klukas, and Falk Schreiber. VANTED: A system for advanced data analysis and visualization in the context of biological networks. *BMC Bioinformatics*, 7(1):109, 2006.

[44] Daniel Jupiter, Hailin Chen, and Vincent VanBuren. STARNET 2: A Web-based tool for accelerating discovery of gene regulatory networks using microarray co-expression data. *BMC Bioinformatics*, 10(1):332, 2009.

[45] Graciously Kharumnuid and Swarup Roy. Tools for in-silico reconstruction and visualization of gene regulatory networks (grn). In *Advances in Computing and Communication Engineering (ICACCE), 2015 Second International Conference on*, pp. 421–426. IEEE, 2015.

[46] Wei-Po Lee and Wen-Shyong Tzou. Computational methods for discovering gene networks from expression data. *Briefings in Bioinformatics*, 10(4):408–423, 2009.

[47] Song Li, Masashi Yamada, Xinwei Han, Uwe Ohler, and Philip N. Benfey. High-resolution expression map of the *Arabidopsis* root reveals alternative splicing and lincRNA regulation. *Developmental Cell*, 39(4):508–522, 2016.

[48] Yupeng Li and Scott A. Jackson. Gene network reconstruction by integration of prior biological knowledge. *G3: Genes, Genomes, Genetics*, pp. 1075–1079, 2015.

[49] Zhi-Ping Liu. Reverse engineering of genome-wide gene regulatory networks from gene expression data. *Current Genomics*, 16(1):3–22, 2015.

[50] Kenneth Lo, Adrian E. Raftery, Kenneth M. Dombek, Jun Zhu, Eric E. Schadt, Roger E. Bumgarner, and Ka Y. Yeung. Integrating external biological knowledge in the construction of regulatory networks from time-series expression data. *BMC Systems Biology*, 6(1):101, 2012.

[51] Aviv Madar, Alex Greenfield, Eric Vanden-Eijnden, and Richard Bonneau. DREAM3: Network inference using dynamic context likelihood of relatedness and the Inferelator. *PloS One*, 5(3):e9803, 2010.

[52] Daniel Marbach, James C. Costello, Robert Kueffner, Nicole M. Vega, Robert J. Prill, Diogo M. Camacho, Kyle R. Allison, Manolis Kellis, James J. Collins, Gustavo Stolovitzky, and DREAM5 Consortium. Wisdom of crowds for robust gene network inference. *Nature Methods*, 9(8):796–804, 2012.

[53] Alexander Martin, Maria E. Ochagavia, Laya C. Rabasa, Jamilet Miranda, Jorge Fernandez-de Cossio, and Ricardo Bringas. BisoGenet: A new tool for gene network building, visualization and analysis. *BMC Bioinformatics*, 11(1):91, 2010.

[54] Robert G. Martin and Judah L. Rosner. Promoter discrimination at class I MarA regulon promoters mediated by glutamic acid 89 of the MarA transcriptional activator of *Escherichia coli*. *Journal of Bacteriology*, 193(2):506–515, 2011.

[55] Natalia J. Martinez and Albertha J.M. Walhout. The interplay between transcription factors and microRNAs in genome-scale regulatory networks. *Bioessays*, 31(4):435–445, 2009.

[56] Alejandra Medina-Rivera, Matthieu Defrance, Olivier Sand, Carl Herrmann, Jaime A. Castro-Mondragon, Jeremy Delerce, Sébastien Jaeger, Christophe Blanchet, Pierre Vincens, Christophe Caron, et al. RSAT 2015: Regulatory sequence analysis tools. *Nucleic Acids Research*, pp. W50–W56, 2015.

[57] Victoria Moignard, Steven Woodhouse, Laleh Haghverdi, Andrew J. Lilly, Yosuke Tanaka, Adam C. Wilkinson, Florian Buettner, Iain C. Macaulay, Wajid Jawaid, Evangelia Diamanti, et al. Decoding the regulatory network of early blood development from single-cell gene expression measurements. *Nature Biotechnology*, 33(3):269–276, 2015.

[58] Sara Mostafavi, Debajyoti Ray, David Warde-Farley, Chris Grouios, and Quaid Morris. GeneMANIA: A real-time multiple association network integration algorithm for predicting gene function. *Genome Biology*, 9(1):S4, 2008.

[59] Tamara Munzner. *Visualization Analysis and Design*. CRC Press, 2014.

[60] Kevin Murphy, Saira Mian, et al. Modelling gene expression data using dynamic Bayesian networks. Technical report, Computer Science Division, University of California, Berkeley, CA, 1999.

[61] Jimmy Omony. Biological network inference: A review of methods and assessment of tools and techniques. *Annual Research and Review in Bioliology*, 4:577–601, 2014.

[62] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[63] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[64] Christopher A. Penfold and David L. Wild. How to infer gene networks from expression profiles, revisited. *Interface Focus*, 1(6):857–870, 2011.

[65] Bruno-Edouard Perrin, Liva Ralaivola, Aurelien Mazurie, Samuele Bottani, Jacques Mallet, and Florence d'Alche Buc. Gene networks inference using dynamic Bayesian networks. *Bioinformatics*, 19(suppl 2):ii138–ii148, 2003.

[66] Jimena Ruiz, Ina Haneburger, and Kirsten Jung. Identification of ArgP and Lrp as transcriptional regulators of lysP, the gene encoding the specific lysine permease of *Escherichia coli*. *Journal of Bacteriology*, 193(10):2536–2548, 2011.

[67] Robert Sablowski. Plant stem cell niches: From signalling to execution. *Current Opinion in Plant Biology*, 14(1):4–9, 2011.

[68] Purvi Saraiya, Chris North, and Karen Duca. An insight-based methodology for evaluating bioinformatics visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 11(4):443–456, 2005.

[69] 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.

[70] Reut Shalgi, Ran Brosh, Moshe Oren, Yitzhak Pilpel, and Varda Rotter. Coupling transcriptional and post-transcriptional miRNA regulation in the control of cell fate. *Aging (Albany NY)*, 1(9):762–770, 2009.

[71] Paul Shannon, Andrew Markiel, Owen Ozier, Nitin S. Baliga, Jonathan T. Wang, Daniel Ramage, Nada Amin, Benno Schwikowski, and Trey Ideker. Cytoscape: A software environment for integrated models of biomolecular interaction networks. *Genome Research*, 13(11):2498–2504, 2003.

[72] Michael E Smoot, Keiichiro Ono, Johannes Ruscheinski, Peng-Liang Wang, and Trey Ideker. Cytoscape 2.8: New features for data integration and network visualization. *Bioinformatics*, 27(3):431–432, 2011.

[73] Matthew E. Studham, Andreas Tjärnberg, Torbjörn E.M. Nordling, Sven Nelander, and Erik L.L. Sonnhammer. Functional association networks as priors for gene regulatory network inference. *Bioinformatics*, 30(12):i130–i138, 2014.

[74] Matthew Suderman and Michael Hallett. Tools for visually exploring biological networks. *Bioinformatics*, 23(20):2651–2659, 2007.

[75] Mehmet Tan, Mohammed Alshalalfa, Reda Alhajj, and Faruk Polat. Influence of prior knowledge in constraint-based learning of gene regulatory networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 8(1):130–142, 2011.

[76] Mingyi Wang, Jerome Verdier, Vagner A. Benedito, Yuhong Tang, Jeremy D. Murray, Yinbing Ge, Jörg D. Becker, Helena Carvalho, Christian Rogers, Michael Udvardi, et al. LegumeGRN: A gene regulatory network prediction server for functional and comparative studies. *PloS One*, 8(7):e67434, 2013.

[77] Y.X. Rachel Wang and Haiyan Huang. Review on statistical methods for gene network reconstruction using expression data. *Journal of Theoretical Biology*, 362:53–61, 2014.

[78] Zixing Wang, Wenlong Xu, F Anthony San Lucas, and Yin Liu. Incorporating prior knowledge into gene network study. *Bioinformatics*, 29(20):2633–2640, 2013.

[79] David Warde-Farley, Sylva L. Donaldson, Ovi Comes, Khalid Zuberi, Rashad Badrawi, Pauline Chao, Max Franz, Chris Grouios, Farzana Kazi, Christian Tannus Lopes, et al. The GeneMANIA prediction server: Biological network integration for gene prioritization and predicting gene function. *Nucleic Acids Research*, 38(suppl 2):W214–W220, 2010.

[80] Adriano V. Werhli, Marco Grzegorczyk, and Dirk Husmeier. Comparative evaluation of reverse engineering gene regulatory networks with relevance networks, graphical gaussian models and Bayesian networks. *Bioinformatics*, 22(20):2523–2531, 2006.

[81] Adriano V Werhli and Dirk Husmeier. Reconstructing gene regulatory networks with Bayesian networks by combining expression data with multiple sources of prior knowledge. *Statistical Applications in Genetics and Molecular Biology*, 6(1), 2007.

[82] Huayong Xu, Hui Yu, Kang Tu, Qianqian Shi, Chaochun Wei, Yuan-Yuan Li, and Yi-Xue Li. cGRNB: A Web server for building combinatorial gene regulatory networks through integrated engineering of seed-matching sequence information and gene expression datasets. *BMC Systems Biology*, 7(2):S7, 2013.

[83] Bowen Yu, Harish Doraiswamy, Xi Chen, Emily Miraldi, Mario Luis Arrieta-Ortiz, Christoph Hafemeister, Aviv Madar, Richard Bonneau, and Claudio T. Silva. Genotet: An Interactive Web-based Visual Exploration Framework to Support Validation of Gene Regulatory Networks. *IEEE Transactions on Visualization and Computer Graphics*, 20(12):1903–1912, 2014.

[84] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

[85] Xiaoru Yuan, Peihong Guo, He Xiao, Hong Zhou, and Huamin Qu. Scattering points in parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1001–1008, 2009.

[86] Hui Zou and Trevor Hastie. Regularization and variable selection via the Elastic Net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

[87] Hui Zou and Hao Helen Zhang. On the adaptive Elastic-Net with a diverging number of parameters. *Annals of Statistics*, 37(4):1733, 2009.

[88] Min Zou and Suzanne D. Conzen. A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics*, 21(1):71–79, 2005.

[89] Khalid Zuberi, Max Franz, Harold Rodriguez, Jason Montojo, Christian Tannus Lopes, Gary D Bader, and Quaid Morris. GeneMANIA prediction server 2013 update. *Nucleic Acids Research*, 41(W1):W115–W122, 2013.