Algorithms for gene regulatory networks reconstruction

Maksimov, N.V.

Award date:
2015

Department of Mathematics and Computer Science
Model Driven Software Engineering W&I / Software Engineering and Technology

Algorithms for gene regulatory networks reconstruction

Master's Thesis

Nikolai Maksimov

Supervisor:
dr. D. Bošnački

Eindhoven, August 31, 2015

Abstract

Gene regulatory network (GRN) inference is a central problem in systems biology, that has been attracting a lot of research efforts recently. Perturbation experiments, in which the activity of some genes is suppressed, are one of the main tools to tackle the problem. Modern technologies allow biologists to collect large amounts of data about perturbation experiments on molecular level. This data can be used to infer the structure of the real biological networks, that can not be accessed directly, which is essential for the understanding of the mechanism of genes interactions. Discovering these interactions is not only interesting from the fundamental point of view, but is also applied in the pharmaceutical industry to create new medicines.

In this work we analyse the previous methods, highlight their advantages and drawbacks, and propose improvements to enhance the effectiveness of GRN inference. In addition to the improvement of the existing techniques, we present REHub (Reverse Engineering with Hubs) - a novel method, distinguishing the influential genes (hubs) and promoting the genes these hubs interact with. Our approach combines statistical and graph algorithmic methods that exploit the scale-free nature of the GRNs, using the clustering techniques. The combination of the state of the art and novel techniques is implemented in the form of an original framework. To evaluate our approach a series of experiments are performed over in silico data.

A de facto standard benchmark for the GRN inference algorithms DREAM4 shows that the REHub-based methods outperform the best state of the art methods for the network reconstruction. The achieved DREAM score is the highest reported in the literature.

Keywords: Gene regulatory networks, Scale-free networks, Reverse Engineering, Hubs, Graphs, Clustering

Contents

Chapter 1

Introduction

This chapter gives a brief description of the Project and includes the formulation of the problem, place of the topic in a current bioinformatics agenda, the project goals, and the approach outline. Finally, it explains the content and the structure of the thesis.

1.1 Motivation for choosing the topic

Gene regulatory network inference is a central problem in systems biology, hence it attracts a lot of researchers from year to year. A *gene regulatory network* (GRN) or *genetic regulatory network* is a collection of DNA segments (genes) in a cell, interacting with each other by the transcription of the messenger RNA (mRNA), which is translated to a protein, capable of influencing the activity of other genes. This activity, also known as *gene expression*, indicates rates at which genes in the network are transcribed into mRNA, which can transitively lead to a new gene regulation [7]. The process of gene interaction is schematically given in Figure 1.1.

While the topology of such networks could be reconstructed, mainly from so called *omics* studies (genomics, proteomics, transcriptomics) in great detail for many organisms [17], it is still incomplete even for the simplest species. The only way to find out the topology is to reverse-engineer GRNs i.e. to identify interactions between the involved players (genes, proteins, etc.) by analysing the data of systematic and controlled perturbation experiments [11].

Modern technologies, such as microarrays experiments and RNA sequencing, allow biologists to perform such perturbational experiments either by *knocking out* (making completely inoperative) or *knocking down* (significantly suppressing the activity) genes and observing the change in the expression levels of other genes. There is a number of methods exploiting these data to infer the underlying causal network, but this network is not reconstructed ideally - there are factors negatively influencing the effectiveness of such methods. These factors are briefly described in the next Section 1.2 and a more thorough analysis is provided in the Chapter 3

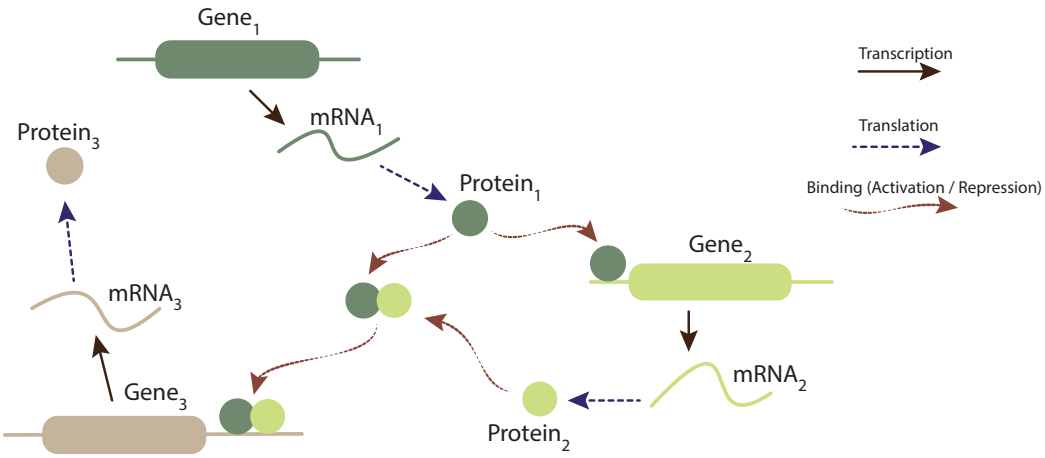

Figure 1.1: Gene interaction: $Gene_1$ regulates $Gene_2$ by the transcription of the mRNA, which is translated to a protein, which serves as a *transcription factor* needed to start the transcription machinery, i.e. influences the rate of the mRNA expression of the $Gene_2$. Other interactions depicted are $Gene_1$ - $Gene_3$ and $Gene_2$ - $Gene_3$.

1.2 Problem description

The problem that is tackled in this work is as follows:

Given results of perturbation experiments on a set of genes, develop a method for reconstruction of the connections between the genes, i.e. the corresponding GRN.

Abstracting from the intermediate steps of mRNA transcription and protein translation, the GRN can be viewed as a directed *graph* $G = (V, E)$, where V is the set of *nodes*, representing *genes*, and $E \subseteq V \times V$ is the set of directed *edges*, representing regulatory *interactions* between them. An edge (i, j) is present in E if and only if a direct causal effect (increase or decrease of the gene expression rate) runs from the node i to the node j.

Two flavors of the problem are *ranking* and *classification*. They are discussed in the Section 1.4, devoted to the approach description. We are mainly focused on the ranking problem, particularly on discovering the real edges, that are undetectable for other methods and promoting them in the ranking list. The ranking list is a list of edges, where the existence of the first edge is most certain and the existence of the last edge is least certain.

Many methods have been developed to this purpose, and a comprehensive evaluation that covers unsupervised (methods that rely on expression data only, e.g., the z-score method), supervised (require information about known interactions for training, e.g., supervised support vector machines (SVMs)) and semi-supervised (the compromise between two) methods, as well as guidelines for their practical application can be found in [13]. The effectiveness of the state of the art methods is not optimal, as they are not

able to solve the following issues, arising when reconstructing the network:

- Perturbation experiments only provide indirect information on gene interaction, thus the effect observed may not always be due to the direct causal effect, but due to the existence of *motifs* - the recurring patterns in a graph that occur more often in GRNs than expected at random. [10]. The most naive approaches are blind to this problem and often recognize a chain effect as a real interaction, thus yielding False Positive results - interactions that do not exist in reality, but are reported by the method. The more advanced algorithms, such as [11] and [18] use so called graph transitive reduction algorithms to remove the edges, wrongly predicted due to this issue. We are using similar techniques as a part of our method.

- GRN structure appears to be neither random nor strictly hierarchical, but *scale-free* [6]. This means that the probability that i regulates k other genes follows a power law, i.e. is $p(k) \approx k^{-\lambda}$, where usually $\lambda \in [2,3]$ [2]. Example of a scale-free network is presented in Figure 1.2.

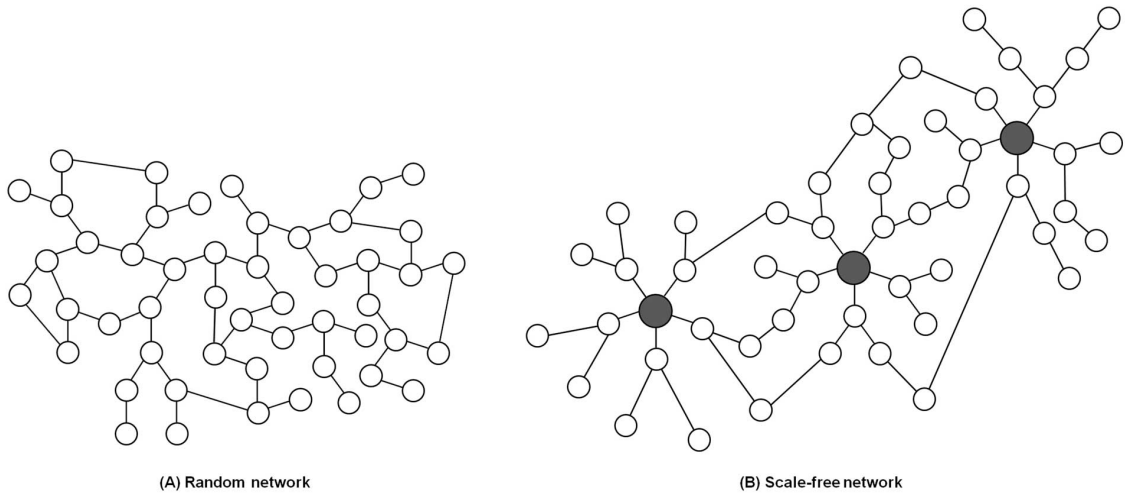

Figure 1.2: Random network versus scale-free network: highlighted are the highest-degree nodes, i.e. hubs. Adapted from [8]

  The main property in a scale-free network is the relative occurrence frequency of vertices with a degree that greatly exceeds the average, i.e. most genes are regulated by only a few others in the GRN. The highest-degree nodes are called *hubs* and though there are very few such nodes, they influence almost all other nodes, hence, one can capitalize on this by developing a method, which treats the hubs in a special way. This fact was noted in [5], but no solution was proposed.

  Another important property of a scale-free network is their hierarchical nature[9]. This allows to search the patterns on different levels and to use such a techniques, as hierarchical clustering.

  We research both of this properties in this work.

- Inferring large scale cellular networks is computationally challenging and only a limited number of computational tools have been developed to address it. Scalability of algorithms is very important, since the ultimate goal is to reconstruct GRNs of full genomes, consisting of more than $10,000$ genes. We solve this issue by preferring the light-weighted methods, such as probabilistic algorithms, over the complex ones, such as supervised machine learning methods.

1.3 Place of the project in the context

Reconstructing gene regulatory networks from high-throughput data is a long-standing challenge. It has been in the bioinformatics agenda for the last 15 years, thus now there is a community of researchers formed around the problem.

One of the challenges in the area is the lack of the gold standards to compare the results of the inference with. Networks can not be extracted from the real organisms directly, and that is the reason to research new inference methods. To be able to evaluate the effectiveness of novel methods, special tools, such as GeneNetWeaver [21] were developed. Using this tool, the in silico experiments can be simulated and datasets, based on the realistic models of the GRNs can be generated.

The Dialogue on Reverse Engineering Assessment and Methods (DREAM) project has been established to intensify the interaction between experiment and theory in the area of cellular network inference and quantitative model building in systems biology [14]. It is organized around annual challenges, where the community of network inference experts is invited to run their algorithms on benchmark data sets, participating teams submit their solutions to the challenge and the submissions are evaluated. These challenges use the datasets, generated with the help of the simulated experiments in GeneNetWeaver, to determine the winner, based on the accuracy of the inference. Though the DREAM4 challenge ended, many works are still using the DREAM4 benchmark to evaluate their algorithms. We are motivated by the same idea, and aim at competing not only with the original contestants, but also with this later "virtual" contestants. We have selected one of the finished challenges, "DREAM4 in-silico size-100", since this challenge takes into account the flaws of the previous challenges and includes the improved experimental datasets to use. The challenge in DREAM4 was to infer network structures from the given in silico gene expression datasets [15]. These datasets contain the gene knock-out and knock-down observations, which is the most mainstream experimental methodology in GRN inference.

1.4 The approach

To develop an effective method we analyse the previous approaches, highlight their advantages and drawbacks, and propose improvements to enhance the effectiveness of GRN inference. The generalized workflow of the inference methods is presented in Figure 1.3.

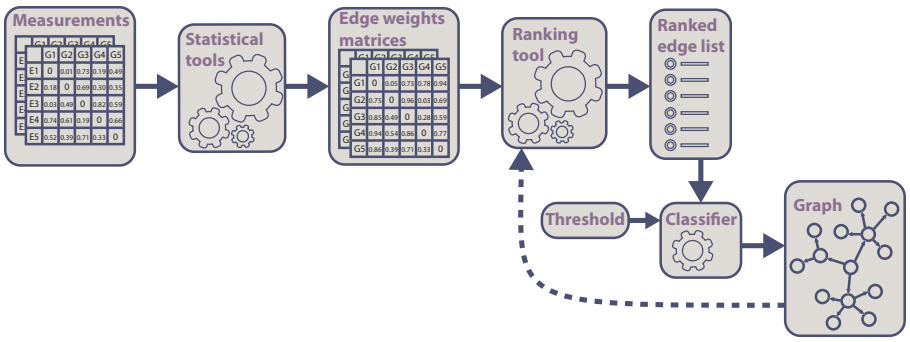

Figure 1.3: Workflow of the network reconstruction methods. Adapted from [5]

The steps of the workflow include:

- The data of the perturbation experiments measurements are received as an input in a form of matrices.
- Statistical tools of choice are used to assign a weight to each edge, possible in the resulting network. This weight represents the estimation of the strength of interaction between the nodes.
- A ranking tool creates a ranked edge list, sorting the weighted edges according to an algorithm. This list can already serve as an output to the DREAM4 framework to evaluate the effectiveness, but to improve the result, the further steps are performed.

- A classifier identifies the edges to include in the predicted GRN graph.
- The graph is transformed to remove and/or add the edges according to an algorithm.
- The ranking tool creates an improved edge list, which serves as an output to the DREAM4 framework.

We are following the same workflow, exploiting a combination of the best working methods from the previous approaches, reported in [11, 18, 5, 4, 12] We analyse these approaches, identify the best elements, and propose the improvements to them.

In addition to the improvement of the existing techniques, we present REHub (Reverse Engineering with Hubs) - a novel method, distinguishing the influential genes (hubs) and promoting the genes these hubs interact with, i.e. increasing their position in the ranking. In the workflow used, REHub is performing a transformation after the graph is generated. The main idea is to incorporate biological motifs in the approach by recognizing certain patterns in a given graph under the assumption of scale-free nature of the underlying network. The previous attempts were focused on one type of the motif - feed forward loop, our work extends this idea and analyses the possibility to distinguish other types of motifs. One of such recurring patterns, namely hubs, is analysed in details and the algorithm, exploiting the knowledge about the hubs existence is proposed. This algorithm is the core of the method, called REHub.

An algorithmic framework was developed to include the REHub to a process of GNR inference. It is based on the available frameworks, but generalizes them, improving such quality attributes as scalability and reusability.

1.5 Structure of the thesis

In Chapter 2 the necessary formal definition of the problem, including the format of the input and of the output data is provided. In Chapter 3 we give a review and analysis of the previous methods. Chapter 4 starts with the analysis of the possibilities to use the scale-free nature of the GRNs, continues with the description of the REHub method and ends with theoretical discussion on generalization of REHub. In Chapter 5 the evaluation method of DREAM is explained, the results of running this evaluation over the REHub method are provided and compared with the results of the state of the art methods. The last Chapter 6 includes the conclusions and the possible directions for the future work.

Chapter 2

Preliminaries

In this chapter the necessary definitions of the problem, including the format of the input and of the output data is provided.

A brief description of gene interaction was already given in the previous Chapter 1 the figure 1.1 was provided, depicting the process of gene regulation, including transcription of the mRNA and translation to a protein.

This chapter also takes into account the underlying biology, but on a higher level, that is relevant to the GRN inference, i.e. the genes are in the scope of this work, but the transcription factors, like proteins and mRNAs, are not.

It contains the problem description, provides general definitions and theorems used in the thesis.

2.1 GRN as Graph

Graph theory is a common way for the GRN reverse engineering community to simplify and display the large amount of relations in genetic networks. Another advantage of this formalization is the potential to apply the techniques developed in other scientific areas, where networks with similar characteristics can be found.

Abstracting from the intermediate steps of mRNA transcription and protein translation, the GRN can be viewed as a directed weighted *graph* $G = (V, E)$, where V is the set of *vertices*, representing *genes*, and $E \subseteq V \times V$ is the set of directed *edges*, representing regulatory *interactions* between them. An edge (i, j) is present in E if and only if a direct causal effect (increase or decrease of the gene expression rate) runs from the node i to the node j. The weight value of the edge measures, how strong is the evidence of the regulatory interaction, i.e. how plausible is the fact that gene i regulates gene j.

Vertex $v \in V$ - A vertex is the fundamental unit of a graph that is linked together to other vertices through edges. In this work a word "node" is used as a synonym, usually in the cases, where it is important to emphasize that a node is a part of a structure.

directed edge - If $(v_1, v_2) \in E$, then we say there is a directed edge from object v_1 to v_2.

directed weighted graph - A directed weighted graph is a tuple (V, E, w) where (V, E) is a directed weighted graph and w is a function $w : E \mapsto \mathbb{R}$. We say that $w(e)$ is the weight of the edge e.

sub-graph - A graph $G' = (E', V')$ is a subgraph of a graph $G = (V, E)$ iff $E' \subseteq E \wedge V' \subseteq V$.

motif - A motif is a recurrent sub-graph of a graph G.

Complete graph G^0 - graph, representing all possible interactions between the nodes.

2.2 Problem definition

Currently, the GRNs can not be extracted from the organisms directly and that is a major problem in the computational systems biology. To cope with this problem the perturbation experiments are performed over the genes, where the genes are perturbed and the measurements of the state of other genes before (also called wild-type) and after are collected.

There are two types of perturbation experiments:

1. *Knock outs* - a genetic technique in which one of an organism's genes is made inoperative ("knocked out" of the organism).
2. *Knock downs* - a genetic technique in which one of an organism's genes is made less operative.

There are several imposed assumptions:

- in each experiment each gene has been perturbed once and the activity of all remaining genes has been measured once after each perturbation.
- the *wild-type* expression level is measured once per gene and for each gene in the network.

There is also an assumption regarding the network, which can be concluded from general biological knowledge:

- the network we reconstruct is sparse, which means that each gene is expected to be connected with only few of other genes in this network.

In such sparse networks a number of nodes called hubs can be identified - they are connected to many others as a source. These hubs are being ignored by the currently used methods, so one of the possible directions of the work is to use their properties, e.g. to identify them and lower the filter to avoid ignoring them and to include the nodes connected to them in the ranking result.

The general problem that is tackled in this work was defined as:

Given results of perturbation experiments on a set of genes, develop a method for reconstruction of the connections between the genes, i.e. the corresponding GRN.

There are two flavors of this problem:

1. *Ranking problem* - Design an algorithm that finds meaningful patterns in gene expression data of gene knock out and knock down experiments. The algorithm must attach a value to the edge of each gene pair in both directions that signifies the certainty of the existence of that edge. The end result will be a ranked list of edges where the first edges existence is most certain and the last edges existence is least certain.
2. *Classification problem* Find a certainty threshold that optimally splits regulatory links into the categories existent and absent. The end result will be the value of a certainty threshold that achieves this and that the edge of each gene pair in both directions is classified as either existent or absent.

2.2.1 Input data

We are using only the data obtained from *steady-state single knockout* and *knockdown* experiments that can be formally defined in the following way:

- The experiment with the set of nodes V, which are *knocked out*, is a tuple (E', x) where $E' = V \times V$ is the set of all possible edges and x is a mapping $x : E' \mapsto \mathbb{R}$

A special notation for indicating the particular types of experiments is used:

- x_{0j} is the *wild-type* (unperturbed state) expression level of gene j, $\forall j \in V$.
- x_{ij}^{KO} is the expression level of gene j after gene i has been *knocked out*, $\forall i, j \in V$.
- x_{ij}^{KD} is the expression level of gene j after gene i has been *knocked down*, $\forall i, j \in V$.

2.2.2 Output data

Network reconstruction method is expected to return the list of all possible regulatory interactions ranked according to the level of certainty in them and (optionally) the graph that represents the reconstructed network.

We formally define them in the following way:

- R is a ranked list, i.e. an ordering of the edges. It can be defined as a mapping of all possible regulatory interactions from edges to positions in a ranked list, such that R_{ij} is the rank of the edge (i, j) assigned by the method, $\forall i, j \in V$. This is a solution of the ranking problem.
- reconstructed network is the graph GP represented as the adjacency matrix A^p such that $A_{ij}^p = 1$ if the edge (i, j) is considered by the reconstruction method as real and $A_{ij}^p = 0$ otherwise, $\forall i, j \in V$. This is a solution of the classification problem.

2.3 Experiments

One of the challenges in the area is the lack of the gold standards to compare the results of the inference with. Networks can not be extracted from the real organisms directly, and that is the reason to research new inference methods. To be able to evaluate the effectiveness of novel methods, special tools, such as GeneNetWeaver [21] were developed. Using this tool, the in silico experiments can be simulated and datasets, based on the realistic models of the GRNs can be generated.

The Dialogue on Reverse Engineering Assessment and Methods (DREAM) project has been established to intensify the interaction between experiment and theory in the area of cellular network inference [14]. It is organized around annual challenges, where the community of network inference experts is invited to run their algorithms on benchmark data sets.

The data provided in DREAM challenges has the fixed format, which is defined in the following way:

- *gold-standard* network is the directed graph GS represented by adjacency matrix A^{gs} such that $A_{ij}^{gs} = 1$ if there is and edge (i, j) in GS and $A_{ij}^{gs} = 0$ otherwise. Matrix row is represented by index $i \in \bar{n}$ and column by index $j \in \bar{n}$, where n is the number of nodes in the network and $\bar{n} = \{1, 2, \ldots n\}$.
- *wild-type* measurements are stored in the one dimensional vector WT such that WT_j is the value of the *wild-type* expression level of the gene j, $\forall j \in \bar{n}$.
- *knockout* measurements are stored in the matrix M^{KO} such that M_{ij}^{KO} is the expression level of gene j after gene i has been *knocked out*, $\forall i, j \in \bar{n}$.
- *knockdown* measurements are stored in the matrix M^{KD} such that M_{ij}^{KD} is the expression level of gene j after gene i has been *knocked down*, $\forall i, j \in \bar{n}$.

Measurement value represents the relative activation of the gene, which is between 0 (the gene is shut off) and 1 (the gene is maximally activated).

Though the DREAM4 challenge ended, many works are still using the DREAM4 benchmark to evaluate their algorithms. We are motivated by the same idea, and aim at competing not only with the original contestants, but also with this later "virtual" contestants. We have selected one of the finished challenges,

”DREAM4 in-silico size-100”, since this challenge takes into account the flaws of the previous challenges and includes the improved experimental datasets to use.

Later in the text ”The gold standard” is used as a shorthand to denote this dataset of 5 networks.

Chapter 3

Methods review

Though, there is an ambiguous terminology in the literature, we are trying to follow the following hierarchy in our text:

1. Approach - is a set of methods;
2. Method - is a set of techniques;
3. Technique - an atomic type of Method;

This chapter contains analysis of the methods present in the state of the art approaches. The best techniques are identified, the flaws of the state of the art approaches are distinguished and improvements are proposed.

3.1 Related works and literature

Many methods have been developed to the purpose of GRN inference in the last 15 years. The attempt to classify all these methods was done in [13] to the following categories:

- *Unsupervised methods* - do not use any data to adjust internal parameters.
- *Supervised methods* - exploit all given data to optimize parameters such as weights or thresholds.
- *Semi-supervised methods* - use only part of the data for parameter optimization, for instance, a subset of known network interactions.

It was noted in [13] that the low prediction accuracies are inherent for unsupervised techniques with the notable exception of the Z-SCORE method on knockout data.

The methods that we are discovering in this chapter are all based on this Z-score method on knockout data. And the novel method we present is also based on the same principles, but is extended with a techniques, which are more characteristic for the methods of machine learning.

One of the first methods of this kind was presented in [19] as a very simple network inference strategy which is called the null-mutant z-score.

It was followed by many others, in the next sections we will only review those of them which gave the ideas to the state of the art best performing method.

The list of common steps of all the algorithms:

- Preprocessing
- Weight assignment
- Edge ranking

- Graph generation
- Graph reduction

3.2 Input data preprocessing / transformation

The simple logarithmic transformation was used in some previous works. It shaped a distribution to better suit the statistical methods, based on the assumption that the underlying distribution of gene expression is normal.

A smarter approach, proposed by [5] worked as following: measure the skewness of M_j values distribution and then transform them in the following way:

$$M_j^{'} = \begin{cases} \left|log(M_j)\right| & \text{if} \quad skewness(M_j) > 0 \\ \left|log(1-M_j)\right| & \text{if} \quad skewness(M_j) < 0 \end{cases} \qquad \forall j \in [1 \ldots n]. \tag{3.1}$$

where $M_j^{'}$ is the column with transformed values corresponding to all measured expression levels of node j, *skewness* is the measure of the asymmetry of the data around the sample mean and $(1-M_j)$ is the operation of making the distribution symmetric to the one observed in M_j. Absolutely symmetric distribution (skewness = 0) is hardly possible, but no transformation is applied in this case.

3.3 Weight assignment

The method used by the best performing approaches follows the same algorithm: combine deviation and correlation statistics, express them as p-values and combine into the single value that represents edge weight. The procedure is similar to the one in [4]:

1. We start from calculating the deviation z-scores, z_{ij}^{KO} and z_{ij}^{KD}, for each edge using both *knockout* and *knockdown* datasets separately and equation

$$z_{ij} = \frac{\left|M_{ij} - mean(M_j)\right|}{std(M_j)}. \tag{3.2}$$

   This technique is illustrated in Figure 3.1.

2. Next they are converted to p-values p_{ij}^{KO} and p_{ij}^{KD} using equation

$$p_{ij} = 2 \cdot (1 - CDF(z_{ij})) \tag{3.3}$$

   and combined to the single p-value p_{ij}^{D} using equation

$$p_{ij}^{D} = CDF(p_{ij}^{KO} \cdot p_{ij}^{KD}) \tag{3.4}$$

3. After that we calculate Pearson correlation coefficient c_{ij} for each pair of nodes using the equation

$$c_{ij} = correlate(M_i, M_j) \tag{3.5}$$

   The equation

$$z_{ij}^{C} = \frac{\left|c_{ij} - mean(C_j)\right|}{std(C_j)} \tag{3.6}$$

   is used to calculate the z-score of each correlation coefficient.

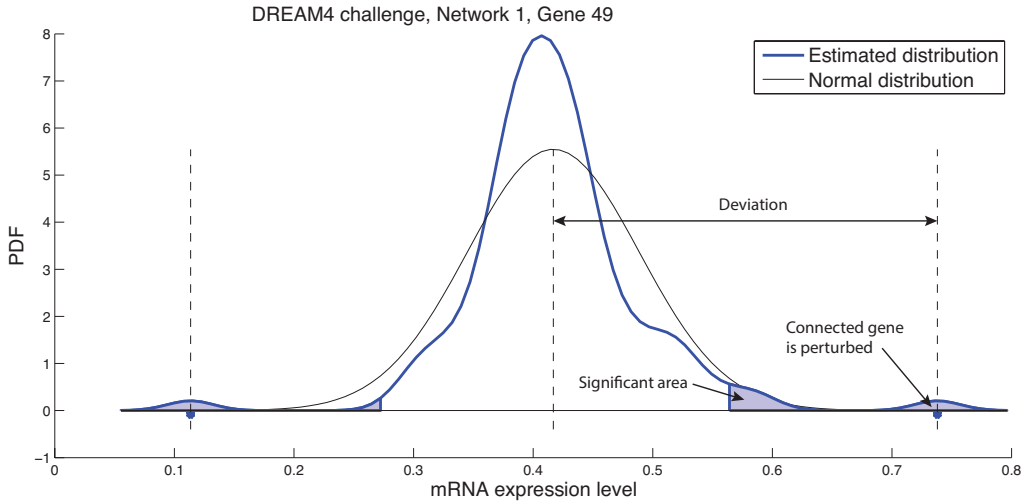

Figure 3.1: Deviation of expression level from its average value. Significant value of the z-score indicates that the perturbed gene probably has influence on the observed one.

4. Obtained values of z_{ij}^C are converted to p-values p_{ij}^C in the same way as in case of deviation (equation 3.3).

5. We assume that correlation and deviation are independent values and calculate the weight of each edge as the p-value of product of p-values of correlation p_{ij}^C and deviation p_{ij}^D using the equation

$$w(i,j) = p'_{ij} = CDF(p_{ij}^C \cdot p_{ij}^D) \tag{3.7}$$

After completion of this step, the following matrices are obtained:

- weights matrix W such that $W_{ij} = w(i,j)$
- deviation p-values matrix P^D such that $P_{ij}^D = p_{ij}^D$
- correlation p-values matrix P^C such that $P_{ij}^C = p_{ij}^C$

3.4 Edge ranking

Since the edge weight is expressed as p-value, we sort the list of all possible edges R according to the assigned weight in the ascending order.

3.5 Graph generation

Thresholds used for edges classification and obtaining the graph G^1 from the initial complete graph G^0 have been defined externally. The process of obtaining the graph representing the reconstructed network can be formalized in the following way:

$$(i,j) \in GP : w(i,j) < 0.01 \quad \forall i,j \in [1 \ldots n] \tag{3.8}$$

where n is the number of nodes in the network to reconstruct. The value 0.01 was selected experimentally.

3.6 Graph reduction

Transitive reduction (TR) is used. The best performing TR is LTR [18]. Threshold values are defined using simple brute force strategy and one of the *gold-standard* networks, which can differ from the ones in the test case. It has been also noted in [18] that algorithms for automatic threshold definition based on actual input data are required.

After the TR algorithm has received an input graph, it checks every edge for being potentially explainable by any indirect path. In case there is such path, the algorithm compares weights of edge and path and makes decision: to remove edge from the input graph or not: an edge (i, j) is not included in the output grapg G^2 if in can be explained by a more certain indirect path. In the LTR method the allowed length of the indirect path is limited by 2, thus this method identifies the triangle patterns, called *FFL* (Feed forward loop). Example of FFL is presented in Figure 3.2.

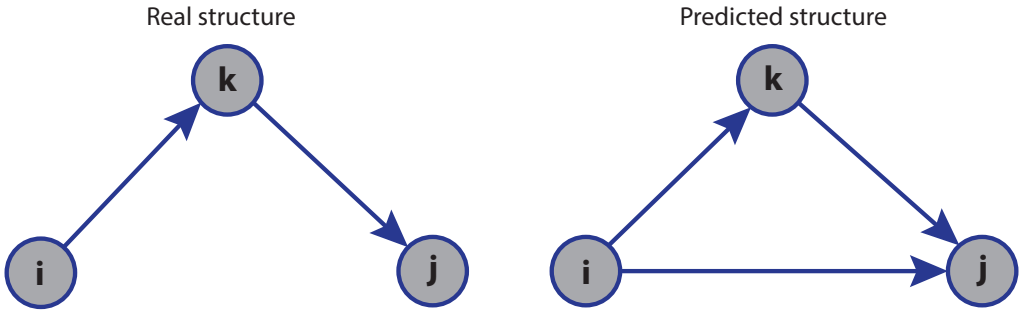

Figure 3.2: Feed Forward Loop.

3.7 Chapter conclusion

We did not go into much details about these methods, since our work only takes the results of these methods as an input. So the key idea of this chapter is to explain, what is the input of our method, i.e. the output of the graph generating methods.

LTR combined with logarithmic transformation shows the best results. So this work takes these methods as a reference and tries to achieve a better scores by improving old algorithms and introducing new.

Chapter 4

REHub

The state of the art methods, based on statistics, are all vulnerable to the fact that the inferred GRNs come from real organisms. The underlying networks are not random, but include *motifs* - the recurring patterns in a graph that occur more often in GRNs than expected at random [10]. There are two main reasons for hubs existence, as stated in [1]:

- Growth: thanks to the growing nature of real networks, older nodes had greater opportunities to acquire links.
- Preferential attachment: as new nodes appear, they tend to connect to the more connected sites, and these popular locations thus acquire more links over time than their less connected neighbors.

These two mechanisms - growth and preferential attachment can be both informally summarized as "rich get richer". From the genetical point of view, it can also be explained as conserving the "richest" structures in the process of evolution: the same motifs are found in different organisms [22].

The best scoring methods, analysed in Chapter 3 implicitly use these facts to find the paths, which better explain the knock-out and knock-down observations, than the direct single edge connection. For example, LTR method, following DR-FFL and TRANSWESD apply the principle of Transitive Reduction to identify and eliminate edges reflecting indirect effects [18], better explained by a certain motif (Feed Forward Loop).

Our idea is to explicitly incorporate biological motifs in the approach by recognizing certain patterns in a given graph. After recognizing these patterns we perform a technique that is to some extent an opposite to the reduction - *edges promotion*.

This chapter starts with a description of the method, able to distinguish and to promote the hub pattern - one of the core structures, found in GRNs and in other biological networks. Section 4.2.3, contains the implementation details of this method.

Section 4.3 describes the Framework, that has been implemented to use REHub with other inference methods.

A theoretical discussion on the possible methods to generalize the method to other patterns completes this chapter in 4.4.

4.1 Definition of the problem

GRN structure appears to be neither random nor strictly hierarchical, but *scale-free*. This means that the probability that gene i regulates k other genes follows a power law, i.e. is $p(k) \approx k^{-\lambda}$, where usually $\lambda \in [2,3]$ [2].

The main property in a scale-free network is the relative occurrence frequency of vertices with a degree that greatly exceeds the average. The highest-degree nodes are called *hubs* and though there are very few such nodes, they are of utmost importance in the properties of the whole network. In GRNs most genes are

regulated by these hubs, so when a hub is found, it could be advantageous to treat it differently, than the regular nodes.

The key idea of the method is *to search for hubs in the graph, constructed during the previous steps of the inference method, and to promote the edges, having these hubs as a source in the final ranking*.

A distribution of the number of edges per node (i.e. a *degree*) in the gold standard is given in Figure 4.1. Here we use the union of five networks from DREAM4 as the gold standard to obtain a larger sample. A histogram of the real distribution can be compared with a curve, plotting the theoretical distribution of degrees, if it followed the aforementioned power law with $\lambda = 2.5$. It can be seen, that the gold standard follows the power law, so it has a scale free structure, which is not surprising, since it was constructed, inspired by the fact that this kind of nets shows the most plausible results when modelling real-life GRNs.[15]

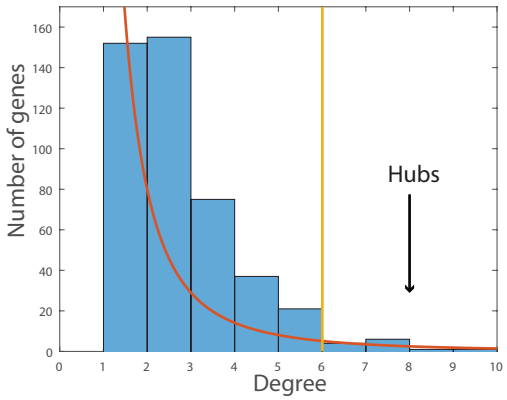

Figure 4.1: Distribution of degrees of the nodes in the challenge.

4.2 REHub algorithm

A vertical line in Figure 4.1 is an example of a threshold, which is needed to distinguish hubs. The definition of hub is not precise, hence there is no single right way to calculate the threshold value. There is a biological cause for this: the major hubs are closely followed by smaller ones. These smaller hubs, in turn, are followed by other nodes with an even smaller degree and so on. This hierarchy allows for a fault tolerant behavior. [23]

4.2.1 General algorithm

We propose the following algorithm to cope with the hubs of different degree.

1. Define a measure of *hub majority*, classifying the node according to its degree, e.g., the hub, influencing 10 vertices is more major, than the hub, influencing 3 vertices;
2. Measure hub majority for each vertex;
3. Apply a function, which, with respect to hub majority of a vertex finds a set of edges to *promote*;
4. Promote edges dynamically changing the intensity of promotion, with respect to the parameter of the hub majority of the influencing vertex.

These algorithms require an additional research to formulate the optimal measures and functions, which is out of the scope of this work. We are implementing only the specific version of this algorithm, which already produces a score that is better than any score reported in the literature. This general algorithm and even more general theoretical discussion in Section 4.4 are provided in this thesis as a fundament for future work.

4.2.2 Specific algorithm

In this section we present a simpler version of REHub, which is based on the ideas, proposed in [3]. It specializes the general algorithm in the following way:

1. The measure of hub majority is boolean, i.e. all the vertices are split into hubs and nonhubs;
2. Promotion is not dynamic, i.e. all the edges in the set are promoted equally.

In Chapter 2 we defined graph as

$$G = (V, E) \tag{4.1}$$

where V is a set of vertices (also called nodes) and E is a set of edges.

Function W maps the edges of the graph to a set of weights:

$$W : E \to [0, 1] \tag{4.2}$$

Further, the notation w_{vu} is used as a shorthand to denote $W((v,u))$

If $(u,v) \in E$, we say that v is a *neighbor* of u. The set of neighbors for a given vertex v is called *neighborhood* of v and is denoted by $\Gamma(v)$.
A *hub* is a node that has many neighbours. In order to classify a node as a hub, it is compared to a threshold τ, which is calculated as an average degree $\bar{d}$ of a node in the graph multiplied to some predefined coefficient γ:

$$\tau = \gamma \bar{d} \tag{4.3}$$

The set of hubs then is defined as:

$$H = \{v \in V | \Gamma(v) \geq \tau\} \tag{4.4}$$

In Chapter 5 we test various values of γ to obtain the best results.

REHub is applied, after the previous methods return an approximation of the GRN, i.e.:

- G^0 - the complete graph;
- W - the specific ranking function, defined on the set of edges of the complete graph G_0, i.e., E_0.
- θ - a threshold, where all the edges (u,v) with $w_{uv} \geq \theta$ are considered to be in the GRN;

When a node is recognized as a hub, we assume that the probability of this node to have more connections, than a regular node, is much higher, so we promote the possible edges that have this hub as a source node. In general, a threshold θ_{hub} is needed to obtain the group of edges to promote. This group then is defined as:

$$E_{hub} = \{(u,v) \in E | u \in H, \theta_{hub} \leq W(u,v) < \theta\}. \tag{4.5}$$

We want to promote this group of edges to make them higher in the resulting ranking, than they were before. Though, they should not be higher, than those edges, which are already believed to be in GRN. To do so, we distinguish two more groups of edges.

E_1 - edges, which are already believed to be in GRN:

$$E_1 = \{(u,v) \in E | w_{uv} \geq \theta\}$$

E_2^{hub} - the least likely candidates edges - not belonging to the graph neither being hub targets:

$$E_2{}^{hub} = E \setminus (E_1 \cup E_{hub})$$

and insert the E_{hub} between these two groups of edges in the new ranking, as presented in Figure 4.2

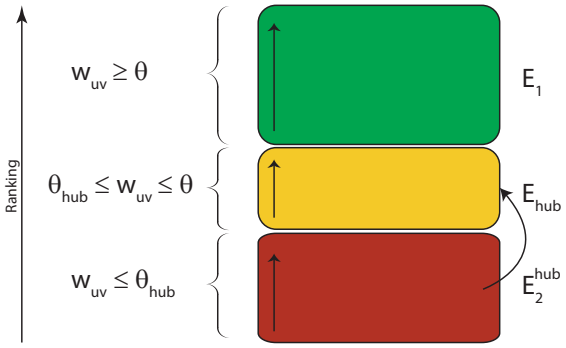

Figure 4.2: Visual representation of the groups of edges, ordered according to REHub method. The arrow illustrates the fact, that G_{hub} consists of elements of G_2, that have been promoted.

It is important to notice, that while performing the promotion, an order inside each group stays unaffected. This allows to save the ordering, obtained during the previous inference methods.

The method is applied after the previous inference techniques where applied for graph construction. Several versions of the graph may be available at this stage, e.g.:

- G^1 - the first approximation of the graph, based solely on the significance measurements;
- G^2 - the second approximation of the graph, after LTR was applied and some edges were removed;
- other graphs G^k - if other types of graph transformation were applied in approach, combining methods.

Which graph to choose for applying REHub is an open question. On one hand, REHub requires a graph as closely describing the reality as possible, since it depends on the properties of the real graph structure. On the other hand, other methods, included in the approach, such as LTR also require the best graph approximation available at the moment.

4.2.3 Implementation of the REHub algorithm

Algorithm 1 shows, how REHub works. The algorithm takes as input adjacency matrix of the graph, which corresponds with the G_1 from the previous section, other parameters have the names, identical to notations from the previous section.

1. A threshold is determined;
2. All the hubs are identified, by comparing nodes degrees (taking into account only significant edges) with the threshold;
3. All the edges, that are outgoing edges of a hub, are compared with the threshold. In case they are, they are collected in a set, that is going to be promoted;
4. The promotion is done by inserting the identified group right after the edges, which are already in the graph;

Algorithm 1 REHub

Require: adjacency matrix A, edge weights matrix W, threshold θ, coefficient γ

{Determine the threshold τ}

$NodeCount = size(A)$

$EdgeCount = sum(A)$ {sum of elements in A}

$d = EdgeCount/NodeCount$ {average degree of the graph}

$\tau = d * \gamma$

{Identify Hubs}

for $i = 1, \ldots n$ **do**

  NodeCount=0 {start counting the neighbors of each node}

  for $j = 1, \ldots n$ **do**

    if $W_{ij} >= \tau$ **then**

      NodeCount = NodeCount + 1

      $G_1 = G_1.put((i, j))$ {Add edge to the list of graph edges}

    else

      $G_2 = G_2.put((i, j))$ {Add edge to the list of not graph edges}

    end if

  end for

  if $NodeCount > \theta$ **then**

    $Hubs.add(i)$ {Recognized hub is added to a set of hubs}

  end if

end for

{Promotion of the edges}

for all h in Hubs **do**

  Edges = edgesOutgoingFrom(h){function extracts all the set of edges, having the hub as a source}

  for $i = 1, \ldots n$ **do**

    for $j = 1, \ldots n$ **do**

      if $W_{ij} < \tau$ **and** (i, j) in Edges **then**

        $G_{hub} = G_{hub}.put((i, j))$ {Add edge to the list of edges to be promoted}

      end if

    end for

  end for

end for

$G_2^{hub} = G_2 \setminus G_{hub}$ {Substract the promoted list from the list, where the edges where before}

Ranking = $G_1 + G_{hub} + G_2^{hub}$ {Concatenate the lists in the new correct order}

return Ranking

4.3 Framework

Ideas, presented in this chapter, have been implemented in Matlab, as a part of an algorithmic framework, which was built, based on the original framework from [18], intended to test the collection of inference scripts. Besides adding the REHub code, the original structure was modified in order to make it more modular and to reuse the code, where possible. The flowchart, representing the Framework is given in Figure 4.3

The steps of the workflow are implemented in the following way:

1. The following data is received as an input:
   - Data of the perturbation experiments measurements in a form of matrices;
   - Settings, including the list of approaches, list of methods in each approach, methods specific parameters, global parameters;
   - Gold standards;
2. Approaches are executed, following this steps:
   - A list of methods for the current approach is determined;
   - The list is executed sequentially;
   - Methods output their results, which can be received as inputs by other methods

   This algorithm allows to combine methods in arbitrary order, without rewriting the code.
3. All classes of methods, including *statistical*, *ranking*, *classification* are run in a standardized manner. They all use the same module, which is controlled by parameters of the concrete method.
4. After all approaches finish and produce the predictions, an evaluation tool (e.g., DREAM4 evaluation script) is run and the evaluation scores are produced.
5. This evaluations scores are now ready to be visualized or compared with each other, or with the scores of the state of the arts inference algorithms.

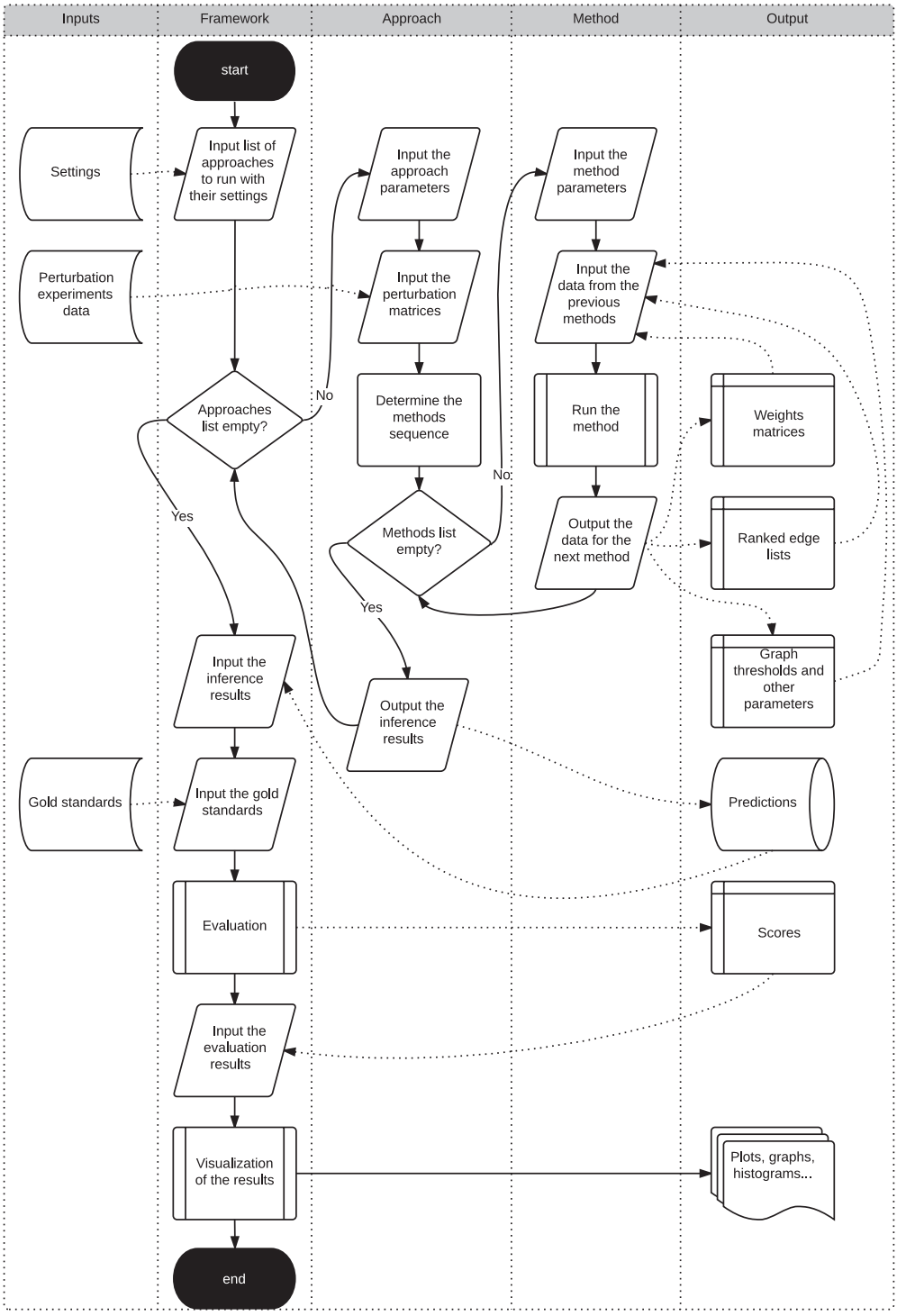

Figure 4.3: A flowchart of the framework.

4.4 Generalization of the method

A potential direction to generalize REhub algorithm is presented in this section, leading to a novel approach of motifs identification using the knowledge about hierarchical structure of a graph.

4.4.1 Biological aspects of scale-free graphs

It is known that interactions between genes have certain structure that can be described by scale-free graphs [16]. Module extraction techniques are used to generate such a graph. Analysis of gene structure shows that hierarchical representation can be used to explain gene dependencies. Figure 4.4 shows the artificial gene network constructed from motif of four-genes structure.

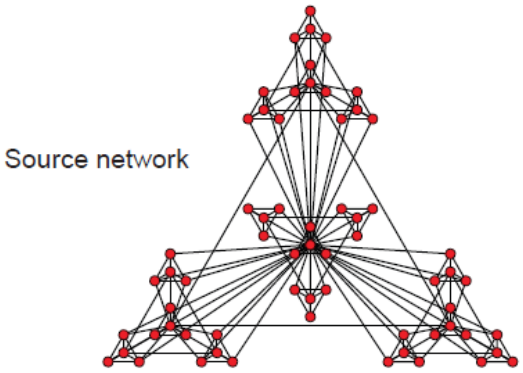

Figure 4.4: Modules of a hierarchical scale-free network.

Under the assumption of the hierarchical structure of graph we want to search for generating motifs on different levels of the hierarchy. It is stated in [15] that the best models of GRNs are based on these generating motifs. That is why extending statistical approaches with this concept is a promising direction for the evolution of the inference approaches of reconstructing the real-life GRNs.

In order to perform this task we propose to explore a hierarchical structure of GRNs. The hierarchical structure can be obtained by various approaches based on pre-knowledge of graph structure, driven by biological experiments or graph analysis and clustering techniques. We concentrate on clustering approach to identify hierarchical graph structure.

The following sections provide a description of the approach that considers GRN as a multilevel structure, where each level is a graph that is constructed from subgraphs, obtained on the previous level.

4.4.2 Generalized REHub algorithm

If a graph is scale-free, then interactions between its nodes are not random, but follow specific patterns.

REhub from the previous sections allows to search for a one particular pattern that is called hub. Generalized version of REHub also searches for particular patterns, but performs it on every layer of the hierarchical structure and also can search for multiple patterns.

The idea of this generalization was proposed by Anna Alperovich (Personal communication, July 2015). We only present a theoretical description of the approach. The implementation is outside of the scope of our project.

The scale-free property of a graph allows us to use its hierarchical structure and to describe the graph in terms of dominant patterns on each layer.

The algorithm must be provided with the *grammar* - a set of possible motifs to look for.

As a result, the algorithm returns a set of generating patterns that determine the graph structure on every level. This sets are subsets of the grammar, each set is a *signature*, which can also be called a *fingerprint*

of a layer. Knowing this signatures allows to choose the specific techniques to use on each layer over each pattern from the signature of this layer.

The algorithm can be described as follows:

1. Cluster initial graph using an agglomerative hierarchical clustering algorithm. By ”agglomerative” we mean: a ”bottom up” approach where each observation starts in its own cluster, and set of clusters that are close to each other are merged into another cluster that moves up the hierarchy. The clustering approach is described in Subsections 4.4.3 and 4.4.4

2. Identify the cluster tree divisions - layers, where motifs (e.g., hubs) will be determined.

3. On each layer starting from the lowest search for the motifs and identify the most represented. The set of the most frequent motifs contains the motifs that, according to the theory [15] are generating the current layer.

4. On each layer starting from the lowest apply the suitable motif-specific techniques, such as edge promotion in case the generating motif is hub.

Figure 4.5 shows an example of a simple graph with hierarchical structure and two patterns.

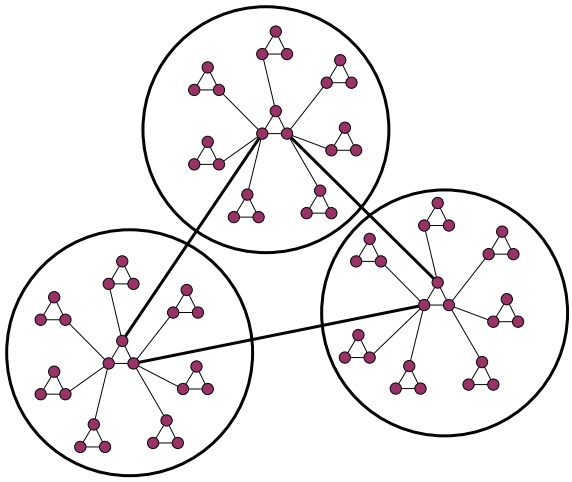

Figure 4.5: Graph with two patterns: hub and feed-forward loop.

The described algorithm, applied to the graph from this figure should work as follows:

1. Consider that the algorithm is provided with a grammar set, including hub motif, FFL motif, and possibly some others.

2. Three layers are identified, intuitively, looking at the picture: the bottom level consists of vertices, the second level consists of triangles as nodes, the top level consists of circles as nodes.

3. The algorithm starts with the bottom layer, where each vertex is a cluster.

4. The set of the most frequent motifs is identified, which consists of only one motif on this layer, which is FFL-triangle. If there were no FFL ”definition” in the grammar, then the algorithm would fail to find any motif on this layer.

5. For each motif in a set a special technique is used, specific for the motif. In this case, FFLs could be downranked with LTR.

6. The algorithm moves to a higher level, where the clusters of vertices from the previous level are taken as nodes.

7. The only motif found is hub. REHub, described in this chapter, can be used to promote the edges, connected to the hub.

8. Then, on the third level, again a new atomic element is considered and an FFL motif of a higher level is recognized. The edges and the nodes of this "meta-FFL" in the figure are highlighted with thick black lines.

We refer to the FFLs at the top level as "meta-FFLs" and the second level hubs can be called "meta-hubs". These "meta-FFLs" and "meta-hubs", could not have been detected with the LTR algorithm and the simpler version of REHub respectively, because they do not consist of vertices, but of groups of vertices.

4.4.3 Graph clustering

In this section, we provide a definition of the graph clustering, that can be used in our desired generalized REHub. The general idea of clustering can be formulated as follows:

Divide a data set into clusters such that the elements assigned to a particular cluster are similar or connected in some predefined way [20].

If the structure of the graph is completely uniform, with the edges evenly distributed over the set of vertices, the clustering computed by any algorithm will be rather arbitrary. Figure 4.6 shows two graphs of the same order and size, one of them is a uniform random graph and the other has a clearly clustered structure.

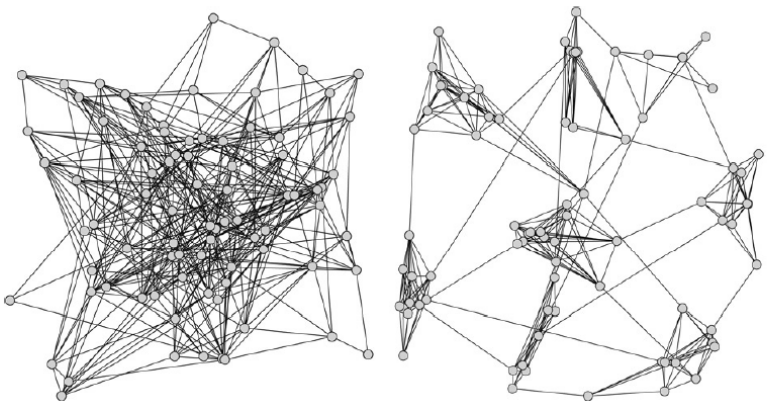

Figure 4.6: Random graph and graph with structure.

To formalize a cluster concept, we provide the properties, which cluster should have:

- Each cluster should intuitively be connected: there should be at least one, preferably several paths connecting each pair of vertices within a cluster.
- If a vertex u cannot be reached from a vertex v, they should not be grouped in the same cluster.
- A metric of distances between vertices must be defined, because only having distances, we can construct clusters.

Measure of similarities based on distances

We present several ways of distance definition in graph theory, for more information see [20].

We start with preliminaries of measure theory where we present how the distance between objects can be defined. A metric or distance measure on a set X is a function (called the distance function or simply a distance)

$$d : X \times X \to R,$$

where R is the set of real numbers, and for all x, y, z in X, the following conditions are satisfied:

1. $d(x,y) = 0$ (non-negativity, or separation axiom)
2. $d(x,y) = 0$ if and only if $x = y$ (coincidence axiom)
3. $d(x,y) = d(y,x)$ (symmetry)
4. $d(x,z) \leq d(x,y) + d(y,z)$ (subadditivity / triangle inequality).

The examples of such distance are Euclidean distance, Manhattan distance.

Our approach requires a way for determining distances between two vertices in a graph. In [20] the definition of such a distance is provided as:

$$d(v,u) = \frac{|\Gamma(v) \cap \Gamma(u)|}{|\Gamma(v) \cup \Gamma(u)|}$$

Having distance between vertices defined we apply hierarchical clustering technique to divide graph into subgraphs.

4.4.4 Hierarchical clustering

A clustering can be defined as a hierarchical structure, where each top-level cluster is composed of subclusters and so forth. This is useful in situations where the graph structure itself is hierarchical, and a single cluster can naturally be composed further to obtain another cluster of clusters. Since GRNs are scale-free, this method is applicable to them.

Clustering methods that produce multilevel clustering are called hierarchical clustering algorithms. A hierarchical clustering is generally constructed by generating a sequence of partitions, where each subcluster belongs to one supercluster in its entity.

In Figure 4.5, which served as an example for 4.4.2 a hierarchical structure is presented, where graph is divided into subgraphs. Each subgraph is considered as a vertex of metagraph, and, again, vertices organized into clusters. The root cluster contains at most all of the data, and each of the leaf clusters contains at least one data element. Such a tree is called a dendrogram; an example is shown in Figure 4.5.

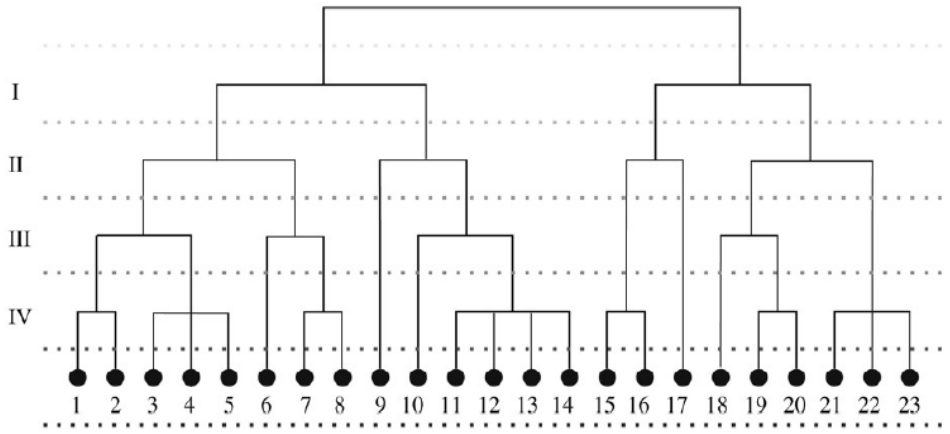

Figure 4.7: Dendrogram example.

4.5 Chapter conclusions

In this chapter we presented our main contributions to the topic:

- We presented REHub - a novel method to improve graphs, derived during the process of GRN inference, by promoting edges, connected to hubs.
- We developed an algorithmic framework, supporting the application of different inference methods and their combinations.
- A generalization of REHub was analysed, including a theoretical discussion on possibilities to improve graphs, derived during the process of GRN inference, by discovering the patterns in a graph.

  The value of this discussion is that it is leading to novel approaches of motifs identification using the knowledge about hierarchical structure of a graph.

Chapter 5

Experiments

An experimental part of the project is given in this chapter. It starts with a description of the standard benchmark for the inference methods. Then the setup of the experiments is given. The chapter ends with the evaluation of achieved results and comparison of this results with the results of the state of the art top-scoring methods.

5.1 DREAM4 benchmark

To quantify the difference between a gold standard and the results of inference methods, DREAM4 uses a special benchmark. Since we are working with the same datasets and want to compare our results with those of other competitors, the same benchmark was chosen to evaluate our results.

To explain, how the score in this benchmark is calculated we first present preliminary definitions of the metrics.

5.1.1 Preliminary definitions

For the in silico data, there is a gold-standard (graph GS) available to compare with the predicted one (graph GP).

- number of *true positive* cases:
$$TP = |GP \cap GS| \tag{5.1}$$
- number of *false positive* cases:
$$FP = |GP \cap GS^c| \tag{5.2}$$
- number of *true negative* cases:
$$TN = |GP^c \cap GS^c| \tag{5.3}$$
- number of *false negative* cases:
$$FN = |GP^c \cap GS| \tag{5.4}$$

where $|GP|$ denotes the number of edges in GP, and GP^c is the complement of GP, i.e. the set of edges: $(i, j) \in G^0 \wedge (i, j) \notin GP \quad \forall i, j \in [1 \dots n]$, where G^0 is the complete graph, and n is the number of nodes. The same metrics can be calculated for GS in a similar way.

In pattern recognition and information retrieval with binary classification, precision and recall are common measurements to evaluate the results.

- *recall*(also known as the true positive rate) – measure of completeness
$$rec = \frac{TP}{|GS|} \tag{5.5}$$

- *precision* – measure of fidelity

$$prec = \frac{TP}{|GP|} \tag{5.6}$$

- *false positive rate* – fraction of negative edges that are incorrectly predicted as positive (also called the false alarm rate, it is complementary to the recall/true positive rate)

$$FPR = \frac{FP}{|GS^c|} \tag{5.7}$$

5.1.2 DREAM score

In order to generalize evaluation, in the DREAM challenges the following approach is used. No specific threshold for the edges from the ranked list to be considered as graph has to be defined by the method, but all the possible ones are tested. The obtained values are then aggregated into a single measure – DREAM score. This evaluation approach includes the following steps:

1. define a set of all possible values of threshold $[1,\ldots,|G^0|]$
2. apply each of the thresholds to the ranked list R and calculate $prec/rec(TPR)/FPR$ measures
3. make two curves by using obtained arrays of values as coordinates

   Receiver Operating Characteristic (*ROC*) curve that graphically explores the trade-off between TPR (y-axis) and FPR (x-axis) as the threshold is varied.

   Precision Recall (*PR*) curve that graphically explores the trade-off between *precision* (y-axis) and *recall* (x-axis) as the threshold is varied.
4. calculate areas under these curves

   $AUROC$ that is a single number that summarizes the trade-off between TPR and FPR.

   $AUPR$ that is a single number that summarizes the precision-recall trade-off.
5. in order to evaluate statistical significance of the $AUROC$ and $AUPR$ values, we compare them with the ones obtained from the random ranking method and calculate 2 corresponding p-values, p_{AUROC} and p_{AUPR}. Distributions for $AUROC$ and $AUPR$ values were estimated from 100,000 instances of random network link permutations.
6. since there are multiple networks in the challenge but the single score value is needed, overall p-values are calculated next as the geometric means of individual p-values corresponding to each network:

$$P_{AUROC} = \left(\prod_{i=1}^{m} p_{AUROC_i}\right)^{\frac{1}{m}} \tag{5.8}$$

$$P_{AUPR} = \left(\prod_{i=1}^{m} p_{AUPR_i}\right)^{\frac{1}{m}} \tag{5.9}$$

   where m is the number of networks in the challenge.
7. finally they are combined into the single DREAM score in the following way:

$$-\log_{10}\sqrt{P_{AUROC} \times P_{AUPR}} \tag{5.10}$$

Simply speaking, we take two ranked lists of edges, from the evaluated method and from the random ranking one, compare them with the real network and assign the score value to the evaluated method. Different methods can be evaluated in this way and ranked according to reconstruction effectiveness. Higher score value corresponds to the better effectiveness of the method.

5.2 Experimental setup

In Chapter 3 the best performing method was identified. It is the method, proposed in [18], which combines z-scores and correlation to calculate edge weights and performs a novel transitive reduction over the generated graph, called LTR. Additional step for this method was proposed in [5] to improve the shape of distributions before executing the statistical tools – the logarithmic transformation.

Later in this chapter the notation LTR_{best} is used as a shorthand to denote the combination of the following steps:

- Simple logarithmic transformation as a preprocessing step.
- Perturbation Graph generation PG^{new} from [18].
- Signed and weighted version of LTR as a transitive reduction step(LTR^{sw} from [18]).

The dataset, used for the experimental inference for the evaluation was defined in Chapter 3 and is used through the report under the name "the gold standard". It is a gold standard from the popular challenge "DREAM4 in-silico size-100", consisting of 5 networks of size 100, which are to be inferred. The DREAM4 benchmark yields a total score as an average score for the predictions for all 5 networks.

5.3 Performance of REHub

5.3.1 External parameters to test

In Chapter 4 we defined two external parameters for REHub:

- A coefficient γ, which is, multiplied by the average degree $\bar{d}$, denotes a threshold for a node to be recognized as a hub:

$$\tau = \gamma \bar{d} \tag{5.11}$$

  The higher this coefficient is, the less number of potential hubs is recognized. The meaningful values for this parameter start from 1. Values less than 1 would imply that hub is a node that has a lower degree, than an average node has, which is contrary to the ideas presented in the previous chapter.

  The highest value for this coefficient also exists - the number of edges is finite, thus there is a 'ceiling' that edge can reach.

  Our aim is to determine this maximal value, and to test all the values of this coefficient from 1 to the maximal value.

- A threshold θ_{hub}, which is needed to obtain the group of edges to promote. We want to promote this group of edges to make them higher in the resulting ranking, than they were before. Though, they should not be higher, than those edges, which are already believed to be in GRN.

  If this threshold's value is 0, then a set of vertices to influence for each hub is the set of all the vertices from all the possible connections from the complete graph G^0, having the hub node as a source. This makes sense, if the number of hubs distinguished is relatively low, as they all can be regarded as major hubs;

  If this threshold's value is 1, then a set of vertices to influence for each hub is very small, most probably even empty. This holds, since θ_{hub} is a threshold for the weights of edges of the complete graph. If the weight of an edge is equal to 1, than in most cases this edge is already in the graph, thus, its vertices are excluded from the potentially promoted.

  Our aim is to test all the values of this threshold from 0 to 1, and to find out the value, producing the best scores.

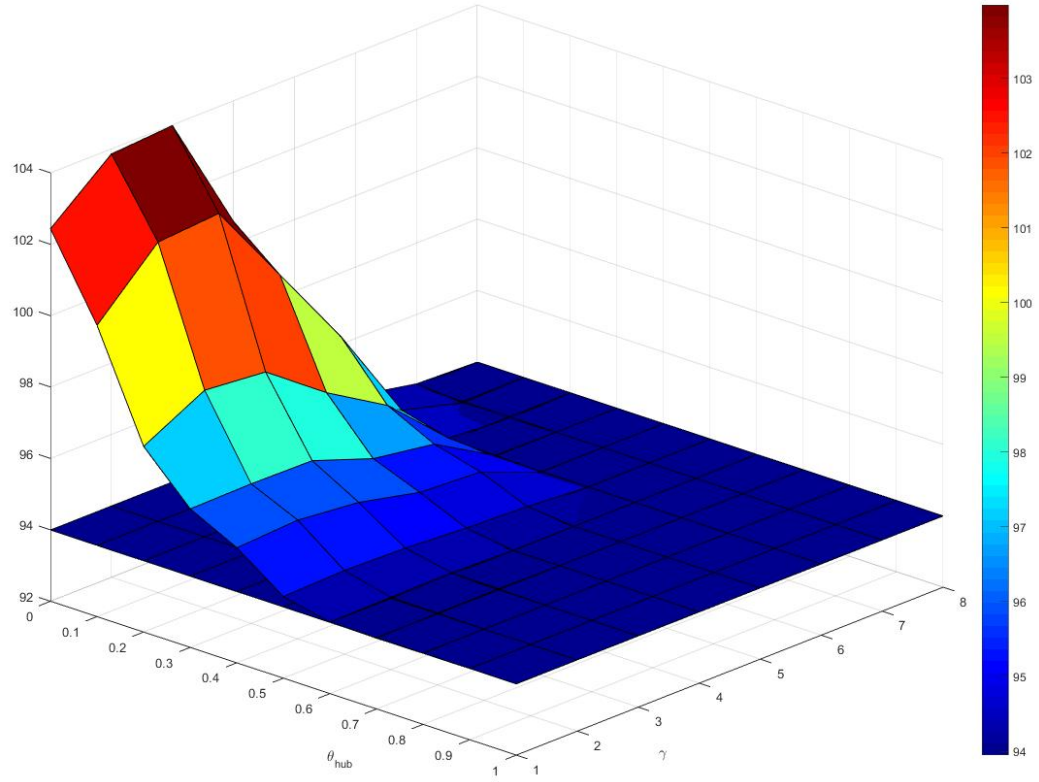

Figure 5.1: Performance of REHub on DREAM4 benchmark with different parameters. Scores of LTR_{best} are also provided.

5.3.2 DREAM4 score

In order to achieve the defined goals, REHub was executed for a set of combinations of intermediate values of the parameters. The result of this is presented in Figure 5.1 The main competitor for REHub is also shown in the figure. It is LTR_{best}, which is not dependent on our parameters, thus is plotted as a flat surface.

5.3.3 Results discussion

After testing the combinations, the best parameters for REHub are found:

$$\gamma = 3.1$$

$$\theta_{hub} = 0$$

The DREAM4 score, achieved with this parameters is **104.2191**, which exceeds the best result, reported in the literature = 93.9955, which was held by LTR_{best}.

The best θ_{hub} value, obtained is 0 and it is constantly decreasing when θ_{hub} is increased. As noted before, the hubs in this implementation of REHub are all major, so they all influence all the possible edges, that have them as a source. Using the general version of REHub, where there are hubs of different majority, could be beneficial in this case.

The best γ value, obtained is 3.1, which gives a threshold τ for a hub equals to 7.02 ($\bar{d} = 2.2655$), so all the nodes with degrees above this level are considered to be hubs.

The fact that our method uses the ranking and the graph, generated by LTR_{best} can explain the fact that the worst scores of REHub in the figure equal the score of LTR_{best}. In the worst cases the set of edges to promote is empty, thus the result of LTR_{best} is not affected.

These worst cases are achieved in two situations:

- $\gamma > 7$ - in this case REHub recognizes no hubs, since the maximum degree in the graph is 15, so there are no edges to promote.
- $\theta_{hub} > 0.4$ - in this case there are no edges, having hubs as a source, which are not in the graph already, so there are no edges to promote.

5.3.4 Area under precision-recall

As stated in [18], the AUPR is the most informative performance measure for the case studies in GRN inference due to the sparsity of gene networks implying large AUROC values differing only insignificantly for the different methods.

In this section we will compare the changes in AUPR before and after applying REHub to prove its contribution to the inference result. In this comparison the REHub with the best parameters, identified in the previous section ($\gamma = 3.1$ and $\theta_{hub} = 0$) is compared with LTR_{best} on the dataset of the network 2 from the gold standard - it was chosen, since the difference in this network is the most significant. The results of this comparison are presented in Figure 5.2 While threshold is low, i.e. a less number of edges is included in the graph, the methods perform equally effective. This behaviour can be explained by the fact that the top ranked edges of REHub are the same edges, that are the edges of the graph in LTR_{best}. Then the curve for LTR_{best} continues to follow the same trajectory, while the curve for REHub not only shows the better performance, than LTR_{best}, but even increases it's own P-R, demonstrated with the lower thresholds. This clearly demonstrate the contribution of the promoted edges to the result.

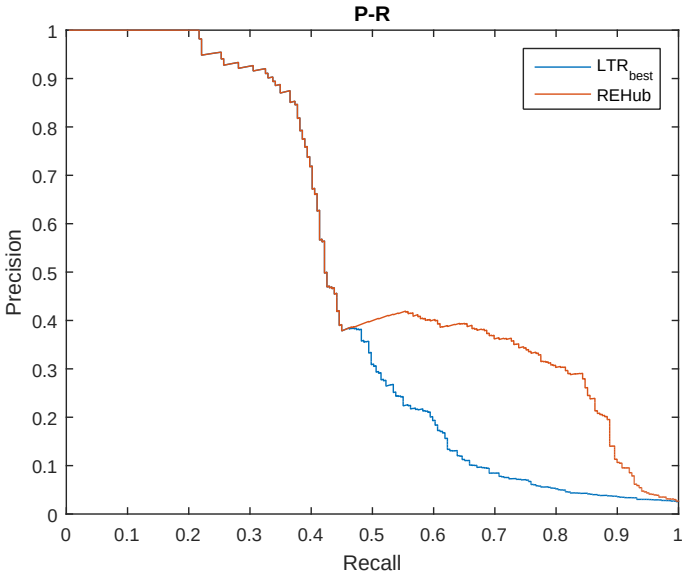

Figure 5.2: Precision-Recall curves for REHub and LTR_{best}

Chapter 6

Conclusions

This chapter reflects back on the Project overall. First it summarizes the work done and our contributions. Then we reflect on the results. The chapter ends with possible directions for the future work.

6.1 Our contributions

Our work and contributions to the topic can be summarized as follows:

- During this project we analysed the state of the arts methods of genetic regulation networks reconstruction and determined, which of the techniques are the best.
- We noticed that even the best performing methods do not use the biological properties of the underlying networks, such as the fact that this networks are scale-free.
- We presented REHub - a novel method to improve graphs, derived during the process of GRN inference, by promoting edges, connected to hubs.
- We developed an algorithmic framework, supporting the application of different inference methods and their combinations, including REHub.
- A generalization of REHub was analysed, including a theoretical discussion on possibilities to improve graphs, derived during the process of GRN inference, by discovering the patterns in a graph. The value of this discussion is that it is leading to novel approaches of motifs identification using the knowledge about hierarchical structure of a graph.

6.2 Reflection on the results

We evaluated the performance of REHub, using the dataset of DREAM4 competition and compared our scores with the state of the art best scoring algorithms. REHub outperforms them all, scoring 104.2191.

To find the reason of this success, apart from the theoretical discussion, presented earlier, we analysed the visualizations of the performance of our method versus the previous leading method. The conclusion is that the edges, promoted by our method are indeed the edges, that were not noticed in the previous approaches.

6.3 Future work

Description of the possible further extension of the Project is provided here.

This work presented novel ideas, many of which need further development:

- We have only experimented with in silico datasets. There are more in silico and even some ”consensus” in vitro datasets to apply our approach to. The ultimate goal of the GRN inference is to discover the real life GRNs, which is also the goal for the future.
- The implemented REHub requires external parameters. We have checked all the possible combinations for the benchmark case, but other datasets may require other parameters. One of the possible directions for the research could be automating the discovery of thresholds. The promising technique for this is a hierarchical clustering, discussed in Section 4.4.4, since it includes the appropriate methods, such as so called *elbow method* - an optimal number of clusters detection algorithm.
- A very promising direction is implementation of the generalized REHub, presented in 4.4.2. Our analysis shows that so called ”meta-motifs”, such as ”meta-hub” and ”meta-FFL” can be found in GRNs, which is currently not done by the state of the arts methods. Implementation of REHub, presented in this work - is a small step to that direction.
- A less complex idea is to combine the REHub with other techniques, not only with LTR. It can be used with graph, generated by any other method. The order, in which the methods are executed is also an important consideration.

6.4 Acknowledgements

I would like to thank my supervisor Dragan Bošnački for all the guidance he provided during my work on this project. I am grateful to the committee members dr. E.P. de Vink and dr. M.A. Westenberg for the professional assessment of my work and valuable remarks, provided during my defense. Also, I would like to thank my parents and friends for their support.

Bibliography

[1] Albert-László Barabasi. Scale-free networks. *Scientific American*, (May), 2003. 15

[2] Albert-László Barabási and Zoltán N Oltvai. Network biology: understanding the cell's functional organization. *Nature reviews. Genetics*, 5(2):101–113, 2004. 3, 15

[3] Kai Bouman. Reverse Engineering of gene regulation. *The internal report, Eindhoven University of Technology*, 2015. 17

[4] B.J.T. Breuer. RegentZDiCo: a method for reverse engineering genetic networks. *Master's Thesis, Eindhoven University of Technology*, 2013. 5, 12

[5] Pavel Dashko. Regina: Reverse engineering algorithm for gene regulatory networks. *Master's Thesis, Eindhoven University of Technology*, dec 2014. 3, 4, 5, 12, 29

[6] Christopher Fogelberg and Vasile Palade. Machine Learning and Genetic Regulatory Networks: A Review and a Roadmap. In *Studies in Computational Intelligence*, volume 201, pages 3–34. Springer Berlin Heidelberg, 2009. 3

[7] Matthew He and Sergey Petoukhov. *Mathematics of Bioinformatics*. John Wiley & Sons, Inc., 2010. 1

[8] Jihyung Lee Buhyun Youn Hyunjeong Seo, Wanyeon Kim. Network-based approaches for anticancer therapy (Review). *International Journal of Oncology*, (6):1737–1744, 2013. 3

[9] JS Kim, K-I Goh, B Kahng, and D Kim. A box-covering algorithm for fractal scaling in scale-free networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 17(2):026116, 2007. 3

[10] Hiroaki Kitano. Computational systems biology. *Nature*, 420(6912):206–210, 2002. 3, 15

[11] Steffen Klamt, Robert J. Flassig, and Kai Sundmacher. TRANSWESD: Inferring cellular networks with transitive reduction. *Bioinformatics*, 26(17):2160–2168, 2010. 1, 3, 5

[12] Willem P A Ligtenberg, Dragan Bošnački, Perry D Moerland, and Peter A J Hilbers. REGENT : Reverse engineering of gene regulation networks using thresholds on correlation and p-values. *Not published*. 5

[13] Stefan R. Maetschke, Piyush B. Madhamshettiwar, Melissa J. Davis, and Mark A. Ragan. Supervised, semi-supervised and unsupervised inference of gene regulatory networks. *Briefings in Bioinformatics*, 15(2):195–211, 2014. 2, 11

[14] Daniel Marbach, James C Costello, Robert Küffner, Nicole M Vega, Robert J Prill, Diogo M Camacho, Kyle R Allison, Andrej Aderhold, Kyle R Allison, Richard Bonneau, Diogo M Camacho, Yukun Chen, James J Collins, Francesca Cordero, James C Costello, Martin Crane, Frank Dondelinger, Mathias Drton, Roberto Esposito, Rina Foygel, Alberto de la Fuente, Jan Gertheiss, Pierre Geurts, Alex Greenfield, Marco Grzegorczyk, Anne-Claire Haury, Benjamin Holmes, Torsten Hothorn, Dirk Husmeier, Vân Anh Huynh-Thu, Alexandre Irrthum, Manolis Kellis, Guy Karlebach, Robert Küffner, Sophie Lèbre, Vincenzo De Leo, Aviv Madar, Subramani Mani, Daniel Marbach,

Fantine Mordelet, Harry Ostrer, Zhengyu Ouyang, Ravi Pandya, Tobias Petri, Andrea Pinna, Christopher S Poultney, Robert J Prill, Serena Rezny, Heather J Ruskin, Yvan Saeys, Ron Shamir, Alina Sîrbu, Mingzhou Song, Nicola Soranzo, Alexander Statnikov, Gustavo Stolovitzky, Nicci Vega, Paola Vera-Licona, Jean-Philippe Vert, Alessia Visconti, Haizhou Wang, Louis Wehenkel, Lukas Windhager, Yang Zhang, Ralf Zimmer, Manolis Kellis, James J Collins, and Gustavo Stolovitzky. Wisdom of crowds for robust gene network inference. *Nature Methods*, 9(8):796–804, 2012. 4, 9

[15] Daniel Marbach, Thomas Schaffter, Dario Floreano, Robert J Prill, and Gustavo Stolovitzky. The DREAM4 In-silico Network Challenge. *Training*, 2009. 4, 16, 22, 23

[16] Miguel Nicolau and Marc Schoenauer. On the evolution of scale-free topologies with a gene regulatory network model. *BioSystems*, 98(3):137–148, 2009. 22

[17] Matthew A Oberhardt, Bernhard ØPalsson, and Jason A Papin. Applications of genome-scale metabolic reconstructions. *Molecular systems biology*, 5:320, 2009. 1

[18] Andrea Pinna, Sandra Heise, Robert J Flassig, Alberto De Fuente, and Steffen Klamt. Reconstruction of large-scale regulatory networks based on perturbation graphs and transitive reduction : improved methods and their evaluation. *BMC systems biology*, 2013. 3, 5, 14, 15, 20, 29, 31

[19] Robert J. Prill, Daniel Marbach, Julio Saez-Rodriguez, Peter K. Sorger, Leonidas G. Alexopoulos, Xiaowei Xue, Neil D. Clarke, Gregoire Altan-Bonnet, and Gustavo Stolovitzky. Towards a rigorous assessment of systems biology models: The dream3 challenges. *PLoS ONE*, 5(2):e9202, 02 2010. 11

[20] Satu Elisa Schaeffer. Graph clustering. *Computer Science Review*, 1(1):27–64, 2007. 24, 25

[21] 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. 4, 9

[22] Lisa Schramm, Vander Valente Martins, Yaochu Jin, Bernhard Sendhoff, and Av Prof L Gualberto. Analysis of Gene Regulatory Network Motifs in Evolutionary Development of Multicellular Organisms. *Artificial Life XII. Proceedings of the 12th International Conference on the Synthesis and Simulation of Living Systems*, pages 133–140, 2010. 15

[23] Bruce J West and Paolo Grigolini. *Complex webs: anticipating the improbable*. Cambridge University Press, 2010. 16