Regulatory Network Discovery Using Heuristics

Armita Zarnegar

A thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

Graduate School of Information Technology and Mathematical Sciences

University of Ballarat

October 2010

This thesis is dedicated to Rosha, my cute 4-year-old daughter.

The time I spent on this thesis belongs to her.

Statement of Originality

Except where explicit references are made, the text of this thesis contains no material published elsewhere or extracted in whole or in part from any other thesis, by which I have qualified for or have been awarded another degree or diploma. No other person's work has been relied upon or used without acknowledgement in the main text and bibliography of the thesis.

Abstract

A Gene Regulatory Network (GRN) is a graph that represents the way in which genes inhibit or activate other genes. The discovery of GRNs is one of the most important and challenging tasks in bioinformatics. This is not only because of the role of GRNs in providing insight into processes and functions inside cells, but also because of their potential for the treatment of diseases and drug discovery. Usually, the technology used for collecting information about changes in gene activity is the *microarray*. Microarray data is complex and noisy and its analysis requires the assistance of computational methods.

This thesis focuses on the automated discovery of GRNs from microarray gene expression data using heuristics from the molecular biology domain. We employed heuristic information for GRN discovery in three different approaches and employed a synthetic data generator called SynTReN to generate different benchmark problems to evaluate each approach.

In the first approach, a combination of local search with gene expression programming was advanced, which we called *Memetic Gene Expression Programming*, to solve a system of differential equations that modelled a GRN. This resulted in an improvement over techniques previously applied to this problem. Our memetic gene expression programming technique also proved to be promising for any other application where there is a need for solving a system of differential equations. Despite the improvements, this method was found to be unsuitable to solve a large-scale real-sized GRN.

In the second approach we used a coarse-grain equation-free model with another combined evolutionary algorithm (*Memetic Algorithm*) for the automated discovery of large scale real-sized networks. In this approach, we found that the evolutionary algorithm was not sufficiently efficient for exploring such a large search space.

In the third approach, we integrated heuristics from domain knowledge to a greater extent than the two previous approaches. The third approach followed two strands. In the

first strand, we advanced a new method to measure and visualize the way a gene activates or inhibits another gene. We called this a *2D Visualized Co-regulation function* and used it to select gene pairs for building a GRN. We also advanced two post-processing steps in order to reduce the number of incorrect associations. The first post-processing method used heuristic information and the second one used an information processing technique.

In the second strand, the structural properties of known networks were used to discover the GRN. Finding the correct structure of the GRN has been reported to be the most challenging aspect of GRN discovery. Our solution to finding the correct structure of the GRN is based on using *Hub Network* to build the core structure of the network. Hubs are nodes with a high number of links attached to them and are known to be the most important genes. We first detected hub genes from domain knowledge and then built a network based on them from microarray data. This resulted in a plausible structure for building the rest of the network. We built the rest of the network incrementally using heuristic information such as the degree of the nodes.

The results obtained using the third approach showed considerable improvement in the performance of GRN discovery when we compared them against existing approaches. We thus demonstrated that the process of discovering GRNs can be improved by using heuristic information along with computational modelling.

Acknowledgments

"Every blade of grass has its Angel that bends over it and whispers "grow, grow."

-The Talmud

A PhD is a journey of exploring which gives us the knowledge of how to conduct research. Supervisors are the lighthouses showing us the path. Once someone asked me, "What is the big achievement of your PhD?" The answer for that is, a PhD is a journey not a destination. I believe the big achievement is the path itself, which gives us the ability, ultimately, to be an independent researcher.

A PhD is a journey toward discovering yourself as well. On this path we not only learn so much about research but also about ourselves. It challenges us in those areas where we have always had problems. Andrew, my associate supervisor, told me this when I began my journey. I was blessed to have him as the lighthouse. I learnt valuable life lessons from him that I will never forget. As an excellent researcher, he also taught me how to conduct good research. I am also grateful to my principal supervisor, Peter, who always gave me freedom to follow my path. Our good relationship allowed me to work with both of them perfectly and I received support from both of them whenever I needed it. I am sure that I will miss this valuable time with them.

A good travel mate along this journey was Cameron Foale. His familiarity with the programming language was so helpful, especially during the early programming phase of this undertaking and it saved me so much time. He offered this help while he was a full time PhD student and had two other part time jobs to support his family as well. Thanks Cameron.

There are some other people who made a difference in this journey. First of all, I would like to thank Maura McCabe for her unconditional support and help, especially for editing the thesis. I also would like to thank Professor Sidney Morris the Head of School

who created a suitable environment for study and supported me during my PhD, by providing me a scholarship extension and travel opportunities to conferences. Thanks to Diane Clingin from the universities research office, a professional woman who always supports research students and has a good understanding of the nature of research. I benefited from her ideas and support. I also would like to thank researchers in the Ballarat Research Cancer Laboratory: Dr Monir Islam who introduced me to the molecular biology field and Dr. Jenny West for additional support in microarray data analysis. I am also thankful to Dr. France Cheong for his kindly support and advice on evolutionary algorithms techniques. Eran Segal, Tim Van den Bulcke and Shoaib B. Sehgal are humble scientists in another part of the world who answered questions from a PhD student, regardless of how naive they were. I appreciate their time for answering my emails.

I am also grateful for the CIAO scholarship provided by the Centre of Informatics and Applied Optimization of University of Ballarat, University of Ballarat Postgraduate Research Scholarship (UBPRS) provided by the University of Ballarat and financial support by the Graduate School of Information Technology of University of Ballarat.

Finally, I am indebted to my husband, Bahador Ofoghi, for his understanding, patience and support when it was most required. He did all these things while he was a full time PhD student and worked part time. In addition to his support, he was always a good model of a researcher for me. I am also grateful to my parents for planting the seed of valuing literacy and education in me. They have been always an immense source of inspiration. My father, by giving me life lessons and introducing valuable books to me when I was a child and my mother, by being a good example of a hardworking, determined and a high achieving woman. This thesis is dedicated to my four year old daughter Rosha, who is as old as this PhD. The time I have spent on my thesis really belonged to her.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

"The larger the island of knowledge, the longer the shoreline of mystery "

-'Ralph W. Sockman

1.1 Prelude

This thesis explores the broad area of gene regulatory network inference. A gene regulatory network is a network that has genes as elements and edges as relationships between them. To understand this thesis, it is essential to have some basic background in molecular biology. Therefore, here we provide an overview of the basic definitions and mechanisms of molecular biology.

All living things are made of cells. Complex animals such as humans have billions of cells. Each cell has a built-in program which controls its functionality. This program exists in a molecule called deoxyribonucleic acid (DNA). DNA is the most important molecule in the cell, as it has codes for all functionality including reproduction and death.

A functional unit of DNA is a gene. A gene is simply a part of DNA which has a code to create a product which is usually a protein. Each DNA molecule has hundreds and thousands of genes. Each of these genes can produce proteins. Each protein has a specific job or function in the body; for example, some proteins help muscle cells to contract. Certain proteins help one cell to divide into two, while others prevent the cell from dividing too often. Each human cell has about 30,000 genes; each one makes a protein with a unique function (Cantor and Smith 1999).

The hard-coded genetic information of a gene transfers into proteins in two steps. Figure 1-1 represents these steps. In the first step called *transcription* a gene located in a DNA molecule is transcribed into an individual transportable messenger called messenger RNA (mRNA). Each mRNA contains the copy of a gene's information for synthesis of a particular protein (or small number of proteins). The RNA molecule is chemically similar to DNA but unlike DNA, RNA exists in a single-stranded form which is less stable and subject to cellular degradation. The primary function of RNA is to act as a messenger molecule and carry information copied from DNA. In the second step called *translation*, a protein is produced from the RNA code. The amount of RNA (the number of RNA molecules containing a gene's information) indicates the activity of the corresponding gene.

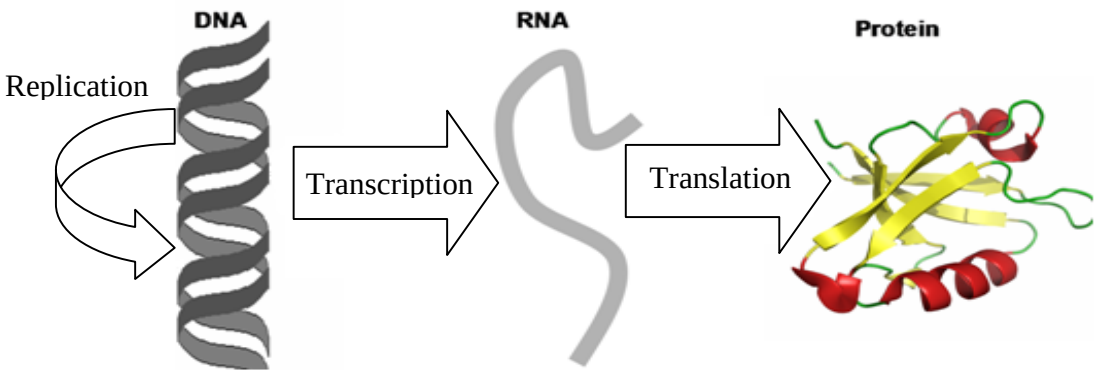


Figure 1-1 Central Dogma of molecular biology

Each cell has a complete copy of DNA but not all the genes are turned on in every tissue. Each cell in our body only expresses a small subset of genes at any time. During development different cells express different sets of genes in a precisely regulated fashion. For example in the presence of lactose in an environment, E. coli starts regulating the gene which produces an enzyme to digest lactose.

Gene regulation or transcription (also called *expression*) is the process by which cells turn the information in a specific gene into gene products. During this process many mRNA copies are produced from the DNA. The number of messenger RNA molecules

(transcripts) available at a given time is called the *gene expression level*. The expression level depends on the rates of transcription and RNA degradation (Kepes 2007).

Each gene is regulated by a certain protein called transcription factors (TFs), which are themselves the product of other genes. Sometimes the expression of a gene requires the function of two or more TFs especially in higher organisms (eukaryotic cells). This means simply that one gene causes the other gene to be expressed which indicates that the processes inside a cell are not stand alone but are connected together to form a sophisticated network. There are connections and communication between processes inside each living cell that makes it possible for these processes to work together and keep a cell alive and active. In molecular biology these networks are called *pathways*. These networks can be categorized based on their function or the type of their elements such as metabolic networks, protein networks, or gene regulatory networks.

A Gene Regulatory Network (GRN) is a network whose nodes are genes and its edges (connections) represent regulatory relationships between genes. These connections can be directional or unidirectional. Also, the connections can be specified in terms of the type of regulatory relationship, such as activation or inhibition relationships. Figure 1-2 represents the gene regulatory network of E. coli described in (2004) which we visualized using Cytoscape 2.7 (Shannon, Markiel et al. 2003).

The ultimate goal of genomics is to understand the genetic causes behind conditions and characteristics. This means having a blueprint that specifies the exact way that genetic components (such as genes and proteins) interact to make a living system. Such a blueprint at a high level is a gene network. Having such a systematic understanding is so critical that it is considered as the second wave in biology and can change how we approach human health (Purnick and Weiss 2009).

In many complex diseases like obesity and diabetes, environmental and other factors contribute along with genetic factors to the creation of the condition over the time (Coleman and Tsongalis 2002). There is usually not a single gene responsible for the condition and usually there is a chain of interactions between genes that result in the

condition. Also, as the processes are all connected it is essential to have a view of the underlying network in order to design treatment.

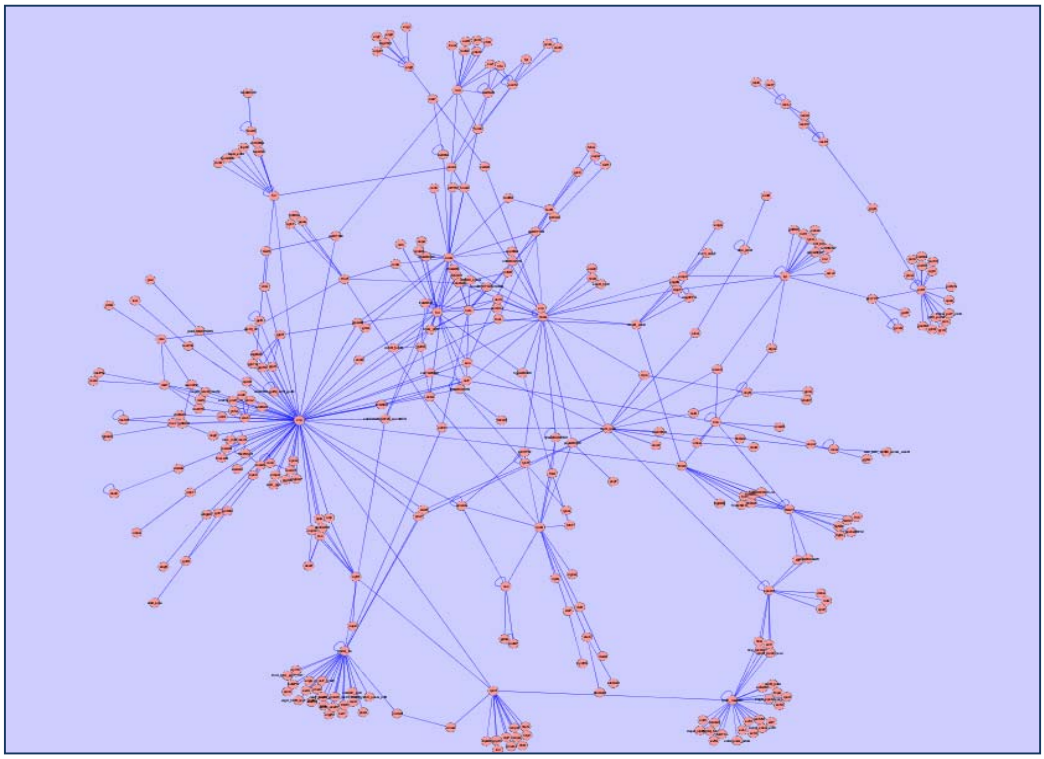


Figure 1-2 E. coli's Gene Regulatory Network

Traditionally, biologists employed laboratory experiments to investigate the cause of a specific condition. Each experiment was usually limited to the study of one gene at a time. Nowadays, with tremendous progress in laboratory technologies it is possible to obtain a big picture and an overall systematic view. As a result, we are faced with a massive amount of data which is hard to make sense of without the assistance of computational methods. There are also other benefits associated with computational modelling. Laboratory experiments are expensive and also time consuming; therefore, researchers use computational modelling and simulation in order to save time and money. Computational modelling provides the freedom to test virtually several types of conditions without bearing the actual costs of the experiments. It enables us to look for all possible combinations of causes simultaneously to find relationships between causes

and effects quickly. This task might be difficult and, in most cases, impossible for a human to carry out as the number of combinations is huge. For example, computational modelling provides the opportunity to measure the dependency of thousands of genes which makes the reverse engineering of gene regulatory networks viable.

Predictive computational models of regulatory networks are expected to benefit several fields. In medicine, mechanisms of diseases that are characterized by dysfunction of regulatory processes can be elucidated (Franke, Bakel et al. 2006). Biotechnological projects can benefit from predictive models that will replace some tedious and costly laboratory experiments. In addition, computational analysis may contribute to basic biological research, for example, by explaining developmental mechanisms or new aspects of the evolutionary process.

The goal of this thesis is to automatically extract the underlying genetic network from the microarray data. Microarray is a technology that enables making a profile of the activity of genes inside a cell (DeRisi, Iyer et al. 1997). This represents the level of activity of each gene inside a cell. Microarray technology has been widely used in a variety of ways in different applications such as cluster analysis for diseases (Eisen, Spellman et al. 1998; Veer, Dai et al. 2002) and functional gene set analysis (Mootha, Lindgren et al. 2003). One popular application of microarray data which has recently emerged is the study of gene regulatory networks. For this purpose, different samples of the same condition have to be captured in different development stages at different time frames (Margolin and Califano 2007). The challenge for computational biology is to discover the underlying pattern or network that the microarray data represents. This is not a trivial task. Several reasons contribute to make this task complex.

First of all, the technology is noisy and there are many biological variations among samples. Secondly, there are usually hundreds of genes or even thousands of them against tens of samples. This causes a problem called the *curse of dimensionality* in data mining. In this situation it is hard to make a robust prediction as there are not sufficient samples compared with the number of features (Tan, Steinbach et al. 2006). Thirdly,

there are variations in the features of a condition over time but the time order of samples in many advanced cells is not obvious. This means that there is no way to find out which sample occurred first and which occurred second and so on, which increases the complexity of the search space dramatically. Moreover, for biologists it is hard and time consuming to interpret the result of computational methods. They have to look at each gene to find its functionality in the molecular biology context and a single gene could have several functions in the body therefore, it is not known which functionality is related to the current situation. The same gene could turn up in many other diseases. In addition, the fact that data is so noisy makes this interpretation task even more complicated. The conclusion is that it is not only hard to analyse microarray data but also even harder to produce a result which makes sense for biologists and is compatible with current knowledge.

There are two facts that led researchers to use the biological domain knowledge along with computational and statistical methods. Firstly, by using and applying domain knowledge there is a greater chance to produce plausible and understandable results that make sense for the biologist. Secondly, existing knowledge can guide the computational method and reduce the complexity. Considering the fact that the nature of the data makes it really difficult to extract a pattern (because of noise and biological variation and not having enough samples compared to the number of features), it is reasonable to use the information from domain knowledge to reduce the complexity.

The pioneering study for incorporating domain knowledge in analysis of microarray data is Gene Set Enrichment Analysis (GSEA) (Subramanian, Tamayo et al. 2005). GSEA looked for gene sets from domain knowledge and tested their significance based on microarray data. Their statistical method was able to find modest changes in a set of genes which derived from Gene Ontology categories. There are several studies that followed GSEA and tried to find changes in the known gene sets based on microarray data (Smyth 2004; Al-Shahrour, Minguez et al. 2006). The aim of the aforementioned studies was the identification of functional gene sets not GRN discovery and therefore they did not consider any dependency or connections between genes.

In the emerging new application of microarray data for GRN discovery, many studies have followed the same path and considered using domain knowledge along with the computational modelling to find networks which are more plausible as well as more accurate. For example, Segal et al (2003) used information about gene set modules in order to partition the search space first and then applied a Bayesian Network to find the dependency of genes inside each module. Other examples are studies by Schadt and his colleagues (Schadt, Stefansson et al. 2008; Zhu, Zhang et al. 2008; Yang, Deignan et al. 2009) which massively used domain information such as position affinity and transcriptional modules.

Currently, methods for GRN discovery still do not have an acceptable performance in practice in the presence of real conditions such as noise, real-sized network or complicated interactions (Marbach, Prillc et al. 2010). There is also a lack of studies related to advanced cells (eukaryotes) such as human cells (except a few such as (Basso, Margolin et al. 2005)). Most of the studies were performed on the simpler eukaryotes cells such as Yeast or prokaryotes such as E. coli. This is because of the difficulty of such a study in more complex cells which have more genetic variations and more genes. Scientists also consider a simple creature such as yeast and E. coli because we already know their regulatory networks and much other associated information(Gama-Castro, Jimenez-Jacinto et al. 2008). It has been demonstrated that we need to use many samples along with the integrated resources from domain knowledge in order to achieve an effective GRN discovery (Zhu, Zhang et al. 2008; Lee and Tzou 2009).

Computational modelling of Gene Regulatory Network has three elements: 1) a dataset containing gene expression measurements; 2) a mathematical model of gene regulation; 3) a search method that can find, within the framework of the model, the networks that are most probable given the dataset and possibly some prior knowledge. These three aspects must be balanced for effective reverse engineering (De Jong 2002; Marbach, Mattiussi et al. 2009).

There are different approaches in computational modelling for GRN inference. Some consider the temporal aspect of gene interactions (dynamic model) while others assume a steady state (static). Usually, the first one is more quantitative and the second one is more qualitative. There is also a deterministic approach that always produces the same answer which contrasts with the stochastic approach. Different techniques have been applied in each of these approaches. Examples of the deterministic approach are Boolean Networks (Kauffman 1969; Sehgal, Gondal et al. 2007; Xiao 2009) and differential equations (Goodwin 1963; Thomas 1981) and an example of the probabilistic approach is the Bayesian Network (Friedman, Linial et al. 2000; Yu, Smith et al. 2004; Liu, Feng et al. 2009). A survey of different modelling approaches for GRNs can be found in Chapter 2.

Our initial aim was to improve the GRN discovery process then we focused on the specific question of how we can balance between using heuristics and domain knowledge and computational methods to achieve a better discovery process. We followed three approaches to answer this question. We will review them in the following section.

1.2 Research Question & Methodology Overview

Our broad research question in this PhD was "*How can reliance on microarray data and heuristics be reconciled to improve GRN discovery?*"

We followed three different approaches in this thesis to answer this question. We started with a quantitative deterministic approach which employed a system of differential equations to model the network. In the second approach, we considered a more qualitative approach which was stochastic in nature. The second approach also incorporated some domain knowledge information to accelerate the search process. The third approach was a deterministic version of the second approach which used heuristics from the outset to limit the search process. The best result was achieved by using the third approach which employed the highest amount of heuristics. This indicated that using heuristics can improve the GRN discovery process. Examples of such heuristics are information about structural properties of GRNs and the nature of co-regulation. The

benchmark datasets for the second and third approaches were produced by a synthetic data generator called SynTReN (Bulcke, Leemput et al. 2006) and the results evaluated based on metrics were previously used to compare performance of well known studies on the SynTReN data (Leemput, Bulcke et al. 2008).

In the following sub-sections we will describe each approach in detail.

1.2.1 Approach1: Memetic Gene Expression Programming

In the first approach, a system of differential equations was used to model a GRN; the problem was considered as a regression problem. In a regression problem, we try to fit an equation or a system of equations to the observed data using different techniques. The current techniques applied to this problem are usually limited to a very small network. We created a novel technique called Memetic Gene Expression Programming in order to solve a system of differential equations which modelled a GRN. We improved on the performance of the current methods by applying Memetic Gene Expression Programming. Our method was compared with previous evolutionary techniques that have been applied to this problem, such as genetic programming, and surpassed them. In addition, the proposed method was tested in the presence of different levels of noise to demonstrate the robustness of the method. Despite achieving an improvement over the current methods, the nature of this type of modelling which requires finding so many parameters made it difficult to scale up to the real size. For this reason, we moved to the second approach, a coarse-grained equation free approach.

In summary, we achieved a successful result; however, the above modelling using differential equations requires precise parameter estimation which makes it difficult for them to scale up to real size. The best method based on this modelling typically can solve only a network of ten genes (Sakamoto and Iba 2001; Kimura, Ide et al. 2004). In reality the number of genes in a network is much greater than this and at least is hundreds of genes. For this reason, in the second approach we moved towards a coarse-grained

combinatorial modelling which is more qualitative than quantitative and can perform under real conditions.

1.2.2 Approach 2: Memetic Algorithm using Domain Knowledge

The second approach provides only the big picture of the network. This approach does not focus on the details and specific parameters of interactions; therefore, it is practical for real-sized networks. In the first approach, we did not use any domain knowledge but in the second approach, we tried to involve domain knowledge in order to accelerate our discovery process. Existing knowledge can provide us with clues to the solution as well as providing a plausible result. Therefore, in the second approach our model took into account some structural properties of GRNs. For example, it considered the modularity of the network and therefore discovered the sub solutions first, and then combined them to find the complete solution (or network). In this approach we also used domain knowledge in the form of gene sets and subnetworks to improve the solutions found by a Genetic Algorithm (GA).

We have noticed, despite several benefits of incorporating domain knowledge, that there is no study in which domain knowledge has been combined with a combinatorial search process like GA. We attempted at filling this gap by proposing a new memetic algorithm (combined GA) which incorporates the information from domain knowledge into a genetic algorithm. In this algorithm, we proposed a new concept of local search that we called *cultural imitation*. In our new memetic algorithm, the evolutionary unit is considered to be a subnetwork and the GA performs the global search process to find the best sub solutions (subnetworks) according to the microarray data. On the other hand, our local search process tries to improve each solution (subnetwork) by replacing it with the most similar solution from domain knowledge by a case retrieval mechanism. In this way, we aimed to speed up the search process and to produce better quality answers in terms of accuracy and compatibility with the domain knowledge network.

The proposed algorithm had an early convergence issue and as a result, its performance was not competitive. The result of this experiment showed us that the evolutionary mechanism was not suitable for such a search space.

Despite the fact that our algorithm did not perform highly in this application, the proposed memetic algorithm has potential and is worth following up in other applications where a simple GA mechanism is applicable. The idea of using a case retrieval mechanism for the local search process may be useful in many applications as we do not need to capture the problem information in a function. Information from domain knowledge in the form of cases can be used directly to improve the solutions by a case retrieval mechanism. The nature of the problem did not let us use a conventional GA and as a result, the proposal that uses the case retrieval mechanism as the local search method was not further studied. The lesson learnt from that experiment was that the search space is too complicated to be explored using just a random search process such as a GA as it could not explore the entire search space effectively. In our third approach, we decided to use more domain knowledge in the form of heuristics to limit the search space from the outset and also to use a guided mechanism by heuristics to explore the search space.

1.2.3 Approach 3: Heuristics Based on Structure and Association Measures

In the third approach, we aimed to use more information and heuristics. In this approach we followed two strands.

First, we explored different association measures for gene-gene relationship. We studied some of the known association measures and found none of them consider the nature of the regulatory relationships. In other words, they look for correlations instead of co-regulation. Therefore, we considered defining an association measure based on the definition of regulatory relationship from a molecular biology context. Our designed co-regulation function has been proved to have a superior performance. For this study we used simulated data produced by SynTReN to produce a variety of regulatory relationships. The tests were conducted in five different types of networks with different

levels of noise and different levels of complicated regulatory relationships. The first version of the function was designed based on a simple one dimensional bin of each gene. The second version was based on a two-dimensional grid with a fixed parameter. The grid was designed to present the relationship between two genes. It has the discretized value of each gene along each axis and the content of each cell represents the frequency of that combination of gene values occurring at the same time in a sample. In the third version of the function we considered having a variable dynamic threshold. The variable threshold function was implemented by two different methods. In the first one we used a sliding window over the grid. In the second method, we built a model using data mining algorithms which was able to learn the patterns across hundred of different samples and apply it to a new search space.

In addition to testing the performance of our designed co-regulation function, we designed other experiments to test our hypothesis about the co-regulation pattern. We used a feature selection algorithm in order to detect the most important features and areas inside the grid. We also applied k-means clustering on the grid to find the most important clusters and then used a decision tree to discover rules which relate the boundaries of these clusters with class labels.

In the following studies, we proposed two ways for post-processing the results of the co-regulation function to improve accuracy. Our first post-processing approach employs heuristic rules in order to eliminate some false positives. In the second post-processing approach, we used a measure of information theory called Data Processing Inequality (DPI) to remove additional false positives. Both of these approaches were useful and improved the results obtained by the co-regulation function.

In the second strand, we analysed the structural properties of GRNs and also patterns, motifs and the other information related to GRN structure in order to find heuristics which could assist us with building the structure of the target GRN more effectively. As a result we arrived at the idea of using *Hub Network* to build the core structure of the GRN. This proposal finds hub nodes from the domain knowledge networks and builds a

network of these hubs based on the expression data. Hub genes are those genes which have more than 4-5 links and usually are the most important and the most essential genes in the network (Goh, Cusick et al. 2007; Barabási, Gulbahce et al. 2011). They hold the network together. We used this initial network as a foundation or primary structure of the network. Our algorithm builds the network structure incrementally and also takes into account the type and the degree of the nodes. In the first step, it builds the Hub Network and then it adds additional nodes from microarray data to this layer. We extracted the degree of each hub from the prior network and used this information to attach nodes to hubs according to this degree. Also, we considered a process of normalization of association measures per gene to select only the most important connection of each gene, which reported a further improvement in performance. The combination of our Hub Network algorithm, our association measure and our post-processing was tested on multiple datasets and networks and achieved excellent results when we compared it with state-of-the-art methods.

1.3 Research Objectives & Contributions

The corresponding research objectives investigated in this thesis and our contributions in solving them are as follows:

- How can we improve the performance of current techniques applied to differential equations modelling of GRNs?
  - We created a new technique called *Memetic Gene Expression Programming* which surpassed the performance of the current techniques (Chapter 5).
- How can we combine gene set information and other information related to genes association in a combinatorial search process (specifically GA) to achieve a better discovery process?
  - We proposed a new Memetic Algorithm which employs a new local search process. The local search process uses a case retrieval mechanism to make the solutions similar to domain knowledge.

On the other hand, the GA performs the global search to find the solutions which match the expression data (Chapter 6)

- How to explain observed gene expression data in terms of co-regulation rather than correlation?
    - Existing association functions in the literature do not provide enough insight into the nature of the gene-gene interaction. They are designed to measure the correlation between any two variables and are not designed specifically to measure genes' co-regulation patterns. We proposed a list of desirable characteristics for an association function to measure gene pairwise relationships. This is presented in Chapter 7.
- How can we measure the association between two genes more precisely compared with the existing functions?
    - We developed a co-regulation function called *2D Visualized Co-regulation function* which looks for the co-regulation patterns (activation and inhibition and dual interactions) and achieved a better performance compared with the common correlation measures used in the literature (Chapter 7 and Chapter 8).
    - Our proposed 2D Visualized Co-regulation function also comes with a visualization ability which makes the result understandable for experts.
- How can we find evidence to prove the assumptions behind our co-regulation function is valid by looking at the microarray data?
    - We validated that by applying data mining techniques on the grid which visualizes the co-regulation. First we transformed the data in such a way that each cell inside the grid represents a feature in the dataset. Then we added class labels such as ac (activation), re (repression) and du (dual interaction) to each record and applied a feature selection process. In addition, we applied the k-means

clustering algorithm to the grid to discover the densest cluster/s inside the grid. We extracted the boundaries of these clusters and formed a dataset which contained boundaries of each cluster, density of each cluster and the class label. In the second step, we used a decision tree algorithm to find the rules which related the boundaries and densities to the class labels (Chapter 8).

Both experiments confirmed our co-regulation assumptions and also opened new questions about the effect of indirect relationship.

- How can we reduce the number of false positives detected by our co-regulation function (indirect relationships)?
  - We proposed a heuristic post-processing operation which looks for the absence of the opposite regulatory relationships in pairs that are already recognized as having a regulatory relationship by our co-regulation function. Once this was applied to the self-regulations only the accuracy was 100% (it did not delete any true answer, just removed the false answers) (Chapter 8).
  - We also applied Data Processing Inequality (DPI) on the output of our co-regulation function. This method successfully increased our performance even further (Chapter 8).
- How can we use the properties of known gene regulatory networks (such as structural properties) in order to design a more effective discovery algorithm?
  - We arrived at the idea of using the *Hub Network* to build the core structure of the GRN. We developed an algorithm which builds a network of the hubs based on the expression data. Our algorithm builds the network structure incrementally and also takes into account the type and the degree of the nodes. In the first step, it builds the Hub Network and then it adds additional nodes from microarray data to this layer. We used a number of links for hubs

extracted from the prior network as a weight of each hub node to guide us in how many edges to consider to be connected to each node. We also considered a process of normalization of association measures for each gene to select only the most important connection of each gene, which reported a further improvement in performance (Chapter 9).

- How can we achieve a better GRN discovery process using heuristic information?
  - We achieved good results by combining our three procedures. The algorithm uses structural properties of the known network (Hub Network algorithm), our proposed co-regulation function and the heuristic-based post-processing procedure. Our results demonstrated that using heuristics we can improve the GRN discovery process (Chapter 9).

1.4 Significance of the Study

This thesis contributes to two different domains of knowledge, computer science and computational biology.

The contributions to computer science are two new techniques that we have developed. The first technique is called *Memetic Gene Expression Programming* which was proposed for the first time in this study. This method is the combination of local search methods with the gene expression programming technique. This technique improves the quality of results produced by using the gene expression programming technique alone and it can be used for a range of problems, especially regression problems. The second technique is a form of memetic algorithm. Our algorithm proposes a new design of GA. Our chromosomes contain a partial solution in the form of a sub graph. Our algorithm also employs a different approach toward the local search mechanism. Our local search uses information from domain knowledge to guide the search space in the form of a case retrieval mechanism.

The contribution of this study to computational biology is the proposal of a new association measure for measuring pairwise dependencies between genes. In this study we first explored the ability of different association measures in the GRN context and identified that the existing measures do not detect the exact pattern of the regulatory relationships. To the best of the author's knowledge, there are not many studies that consider the nature of the regulatory interactions to define an association function. The relationships between gene pairs were considered a black box and researchers have tried different correlations and association measures in order to find the one which achieves the best results.

We proposed a function called *2D Visualized Co-regulation function* based on the definitions of regulatory relationships. Specifically, our association measure is defined based on the definition of inhibition or activation and dual interaction from the molecular biology context. We demonstrated that this function can surpass the previous measures.

Another important aspect of our 2D Visualized Co-regulation function is its visualization power. Most of the correlation or association functions produce a number indicating the degree of association. This does not give much information about the nature of the changes. In contrast, our function not only provides information about the degree and type of the associations between genes but also represents the relationships in a visually powerful and sensible way. This promises to provide insight into the nature of interactions and makes it understandable for molecular biologists.

We also proposed a new heuristic-based post-processing procedure for reducing the number of false positives. The proposed post-processing step considers the absence of the opposite relationship as the indication of true positives; therefore, it removes those which represent some mild patterns of the opposite regulation. Our post-processing step achieved very good performance when it was applied only on self-regulations.

Finally, the last contribution is using *Hub Network*. Specifically, we identified hub genes from the related known network to build a new network. We also used other heuristics such as information about the genes connectivity and the number of their edges to guide

the discovery process and to build the network structure. This information was also used in our proposed background correction process for each gene.

In this thesis, the best result was achieved using heuristics from the outset. We demonstrated the positive benefits of i) using the heuristics related to the structural information of the known network ii) using the definition of regulatory relationships to detect pairwise associations and iii) using heuristics for reducing the number of false positives.

1.5 Overview of Thesis

This dissertation stands at the intersection of computer science and molecular biology. We assumed no primary knowledge in either area and organized this dissertation to provide background in both areas.

- Chapter 2 provides necessary background information in molecular biology as well as the literature related to GRN discovery. This chapter was not meant to be a complete literature review as the related literature to each approach is provided in each chapter separately.
- Chapter 3 describes the difficulty of the benchmarking algorithm for reverse engineering of gene regulatory networks. Consequently, it provides the background and literature about computational benchmarks or synthetic data generators. It then discusses different simulators and the reason why we chose a particular simulator called SynTReN. This is followed by the description of how we generated our six benchmark datasets using SynTReN. At the end of this chapter, we provide an evaluation methodology and performance measures based on the related literature and other studies applied on the data produced by the same synthetic data generator.
- Chapter 4 provides a solid background in the evolutionary computation techniques. This information is necessary for the reader in order to understand our first and second GRN discovery approaches.

- Chapter 5 proposes a novel technique to solve differential equations which model a GRN. The technique is called *Memetic Gene Expression Programming*. This technique was created based on conventional gene expression programming, by adding a local search process to tune the parameters for a system of differential equations more effectively.
- Chapter 6 proposes another novel technique for reverse engineering of GRN, based on a combinatorial search. The search process uses an evolutionary algorithm as the global search process and case-based retrieval as the local search process. The method tries to find the functional subnetworks in the network and to then merge the subnetworks in order to find the overall picture.
- Chapter 7 first provides an overview of our third approach for GRN discovery. It also describes the theory behind the first strand of our third approach which proposes a new association measure for gene pairwise dependency. In doing that, we firstly review the most common association functions for measuring the pairwise dependencies of genes and their assumptions and limitations. We argue that none of them are designed to measure the exact pattern of the regulatory relationships. We then provide a list of the desirable characteristics for an association function, and finally, we describe our designed association measure based on the mentioned characteristics. The experiments with the proposed association measures are described in detail in the following chapter.
- Chapter 8 describes the follow up experiments based on the idea of our association function, proposed in the previous chapter. The experiments with different versions of our co-regulation measure are reported across different synthetic datasets and networks. We started with the basic version of the function and finished with a variable threshold co-regulation function. In this chapter, we also report the experiments with the proposed post-processing techniques.
- Chapter 9 describes the second strand of our third approach. We used the Hub Network idea in order to build the primary structure of the network as well as other background information. We provided experiments to demonstrate the

effectiveness of this idea in terms of improving the performance of current approaches for modelling GRNs. In addition, we report the highest performance of our third approach using the Hub Network idea combined with our co-regulation function.

- Chapter 10 summarizes the key contributions of this dissertation and draws a number of conclusions. It also proposes possible future work and possible further development of our proposed techniques.

Chapter 2

Background

"The gene is the unit of Life. The soul is the unit of Humanity. We know the alphabet of Life, we have unraveled the code. But remember, like words, DNA has significance beyond the sum of its parts."

- David Bromfield

2.1 Introduction

In the introduction chapter we briefly talked about the GRN discovery problem and provided some basic background to the problem in order for the reader to understand the research question. In this chapter we will provide more descriptions and a solid background into the related literature. Extra detailed literature will be provided in the following chapters where we review each of our approaches. We also will provide related literature to evaluation separately in Chapter 3 Experimental Setup.

Here, we first start by reviewing microarray technology and the nature of its data. Then we will briefly talk about the challenges for analysis of such data and different generations of methods and approaches for this purpose. Second, we will describe the nature of Gene Regulatory Networks (GRNs) and gene expression mechanism. We then will talk about the traditional methods for GRN discovery. At the end we talk about approaches towards modelling of GRNs and computational methods which have been developed so far for GRN discovery. This provides enough information in order to understand the approaches was taken in this thesis.

2.2 Microarray Data

Every cell inside a particular living creature contains a complete identical copy of its DNA and identical genes. Only a fraction of these genes are turned on (expressed) in different organisms and conditions. This subset of genes is a unique property of each cell. For example, a particular subset expressed in liver cells is different from the subset expressed in heart cells. Also, in the same cell, genes are expressed differently during different stages and conditions. For example, genes expressed in dehydration are different compared with those under normal conditions.

For many years, biologists studied one gene at a time and it was not possible for them to capture a complete picture of every gene expressed in a cell. Advances in technology enabled them to build a general profile of gene expression, in order to study changes in the cell during different conditions genome-wide. Microarray is a technology to read the state of DNA through the expression of many genes at once (DeRisi, Iyer et al. 1997). It is a tool that allows us to monitor the expression of all genes active in a cell at the same time. A microarray is typically a glass or polymer slide, onto which part of a DNA molecule (usually a gene) is attached at fixed locations, called spots. There may be tens of thousands of spots or genes on an array. Microarray can be used to detect DNA, mRNA, proteins or antibodies. For gene expression studies, we measure gene expression levels by assaying the relative abundance of mRNA molecules, which are the transcribed products of genes in cells. In this research, we are working with mRNA microarray data called gene expression data. To measure gene expression, the mRNA molecules extracted from the target cells are hybridised to a solid substrate with thousands of microscopic spots, each of which contains a certain number of DNA sequences of a specific gene. By measuring the amount of mRNA hybridized to one spot, we can infer the expression levels of the gene in that particular spot. This is due to the assumption that the amount of mRNA is an indicator of the activity of the corresponding gene (DeRisi, Iyer et al. 1997). However, it is not easy to quantify the amount of RNA as representative of a particular gene because this can vary over time. Thus we need to be able to set a

threshold level to specify the presence of a gene. A way to address this is using competitive hybridization experiment. This compares the state of a gene in one condition to that of a reference and allows the gene's expression to be quantified in terms of the changes between the two.

There are different technologies available for microarray chips. Two-colour arrays, single-colour arrays and longitude arrays. Each of these technologies is different in terms of how they produce the gene expression profile. For example, in two-colour arrays after biological experiments and preparing samples, all mRNA extracted from each of the two samples are labelled with two fluorescent dyes; for example a green label for the condition one and a red label for condition two. Then, samples are hybridized and are excited by a laser and scanned. The amount of fluorescence emitted upon laser excitation corresponds to the amount of mRNA bound to each spot. In case the nucleic acid from the sample one is abundant the spot will be green, whereas if sample two is more abundant the spot will be red. If neither is presented, the spot will be black and if the two samples are equally present then the spot will be yellow (Klipp, Herwig et al. 2005). In one-colour array the process is similar, but the main difference is that two conditions are not combined in a single array but in two different chips. Still fluorescence dyes are used to detect the image; however, the calculation of the differential expression of genes is carried out through comparing the intensity of the same spot in different arrays. Examples of one-colour chip are Affymetrix "Gene Chip" and Illumina "Bead Chip".

In general, to perform microarray experiments, the following steps are required: designing and printing of the array; designing of the biological experiment; preparing the samples, sample labelling with fluorescent dyes, hybridization, washing, scanning image acquisition; data cleaning and transformation; background subtraction, normalization; data filtering; and data analysis and interpretation (Dubitzky, Granzow et al. 2003).

These dye arrays are usually prepared and ready for use by the manufacturer and, therefore, the first step is to prepare biological experiments and samples. However, different technologies of microarray such as single-colour or two-colour microarray work

differently, but their output basically shows the same thing which is the differential gene expression profile. The image after the numerical transformation is usually in the form of large matrices of expression levels of genes (rows) under different experimental conditions (columns) with some missing values. After transferring data to numerical values, a pre-processing stage is usually applied to the raw matrices of data in order to remove noise and also replace the missing values. It is estimated that the majority of mRNA from the cell contains transcription from a small minority of genes. The majority of genes are expressed at very low levels (Seidel 2008). Therefore, it is usual that a mammalian sample after analysis has half a probe (spot) as absent because the amount was too low to be detected (Seidel 2008). Therefore, strategies are needed in order to replace the missing values.

After that, a method is required to facilitate comparison between genes by combining the two conditions into one value. The most common way to achieve this is to calculate the average fold ratio of the intensity of the signals in one channel in one condition over the other condition. As this fold ratio can vary hugely depending on the range of genes expression, a log of this ratio is taken to decrease the effect of the difference in the range of the expression and to normalize these changes. The log ratio allows both an increase and a decrease in the expression of a gene to be compared in a unique way (Fraser, Wang et al. 2010). However, the log ratio does not take into account the sample size. It also does not take into account the extent to which the measures vary among samples. Therefore, statisticians use p-value instead to indicate the probability of observing such a difference by chance.

Laboratory techniques related to microarrays have been developed professionally and commercially and are now well established. However, analysis of the results still remains a challenging area (Al-Shahrour, Minguez et al. 2006). The reasons are that the nature of data is highly variable, redundant and noisy.

There are different types of data variability including experimental variation and biological variation. Biological variations happen mainly as a result of different genes in a cell being active at different times. This makes a big difference in samples under study;

however, there is not much that can be done to overcome this. In general, there are two types of biological noise: intrinsic noise and extrinsic noise (Scherer 2009). The intrinsic noise occurs due to the inherent stochasticity of biochemical processes and environmental variations. These make variations in the amount of cellular components, which in turn affect the biochemical reactions. The extrinsic noise occurs due to different developmental stages of cells, e.g., cell cycle variations or continuous mutational evolution.

Biological variability is so high due to individual differences, such as different sex or age (Swain, Elowitz et al. 2002; Purnick and Weiss 2009). This is considerable especially in eukaryotes. Even in a population of genetically identical cells, the response of individuals may be significantly different from the average population response (Novick and Weiner 1957; Elowitz, Levine et al. 2002).

Experimental variation occurs due to the individual sample quality and homogeneity such as RNA degradation and cell heterogeneity in the tissue sample (Scherer 2009). Data variability is sometimes dramatic, which makes it hard to analyse the data, such as cancer conditions where most of the time obtaining information about the type and stage of taken samples is impossible. Data variability also makes it almost impossible to repeat the experiments and produce the same data.

Redundancy occurs sometimes because several copies of a single gene are attached to different probes of the microarray data and this causes confusion about the expression value of a gene.

In addition to data variability and redundancy, noise is often high, which usually makes comparison between platforms and experiments unsatisfactory. It also makes it hard to reproduce the data. A source of noise is non-accurate experiments known as experimental noise. Hofmann (2006) stated that the main reason for noise in microarray data is that there are many experimental steps and, therefore, many sources of data variability. This non-biological experimental variation is known as *batch effect*, which renders the task of data combination from different datasets difficult. A major challenge

in microarray analysis is to effectively dissociate actual gene expression values from experimental noise (Hofmann 2006). One way to overcome such noise is to integrate different data sets related to the same condition. There is software developed to reduce the amount of such experimental noises, an example of which is a package called Limma. Limma is a package developed using the R language as a part of the Bioconductor package (Gentleman, Carey et al. 2004) for differential expression analysis of microarray, which is capable of removing the effect of experimental conditions.

Another challenge of microarray data is that there are usually so many genes compared with the relatively small number of experimental samples, and this causes a phenomenon called the curse of dimensionality. This refers to the phenomenon that makes the data analysis significantly harder as the dimensionality of the data increases (Tan, Steinbach et al. 2006). In such a condition, it is difficult to make any generalizations and extract the pattern from the data. This problem is exacerbated by the fact that microarray data is an expensive technology and it is, therefore, expensive to have many samples and in many research laboratories there are usually less than ten samples available due to experimental limitations.

Statisticians and bioinformaticians have developed algorithms and methods to overcome the difficulty of analysis of the microarray data. Several algorithms have been developed for normalization and transformation, and also for feature selection and feature construction. Normalization and transformation is used to reduce the noise, and feature selection is used to reduce the curse of dimensionality. However, extracting biological information from microarrays still remains a difficult task (Goeman 2007).

Figure 2-1 represents typical microarray data after numerical transformation and pre-processing. Further analysis needs to be carried out for the discovery of the gene groups that are different between two conditions. For example to find out the most effective functional gene sets involved in that condition or ultimately underlying GRNs.

The first generation of tools for analysing microarray data relied only on computational or statistical methods to extract hidden information from the data. The output of such a

system was usually a list of genes which seemed to be different between the two groups of samples; such as diseased versus normal. Alternatively, clustering and classification algorithms were used (mostly hierarchical clustering and K-means) to group genes that had shown a similar pattern of change. The aim was to find different subgroups of patients or different tumour types by clustering genes in groups according to their expression level (Parmigiani, Garrett et al. 2003).

	A	B	C	D	E	F	G
1	GENE	sample_0	sample_1	sample_2	sample_3	sample_4	sample_5
2	lon	0.21658	0.417949	0.220523	0.217615	0.190682	0.26363
3	rpoH	0.086799	0.194768	0.090334	0.0698	0.079782	0.107903
4	mopA	0.205586	0.585748	0.233863	0.303211	0.225044	0.289096
5	htpG	0.153639	0.364902	0.195492	0.138948	0.175024	0.220283
6	rpoE_rseA	0.471023	0.458	0.289723	0.697778	0.784589	0.311621
7	ibpAB	0.198035	0.346645	0.219616	0.205859	0.198929	0.302592
8	ecfABC	0.738185	0.743969	0.547324	0.883583	0.758563	0.446313
9	ecfH	0.629231	0.732528	0.577749	0.835472	0.874402	0.546105
10	fkpA	0.824654	0.736182	0.479968	0.896465	0.857903	0.6058
11	crp	0.612618	0.211149	0.833494	0.719413	0.81461	0.18049
12	araE	0.716904	0.415473	0.658599	0.739979	0.688004	0.42186
13	xprB_dsbC	0.844457	0.481751	0.579903	0.93923	0.965068	0.427394
14	lpxDA_fab	0.604744	0.827833	0.615254	0.944061	0.919743	0.573913
15	ppiA	0.125916	0.423638	0.108218	0.067709	0.076015	0.207275
16	dnaA	0.540723	0.679463	0.190033	0.754664	0.554073	0.626461
17	ptsHI_crr	0.894123	0.439679	0.922779	0.583976	0.608463	0.444519
18	cirA	0.4323	0.260499	0.313266	0.371118	0.434563	0.22229
19	cyaA	0.149694	0.399271	0.049503	0.055126	0.085735	0.481918
20	speC	0.693518	0.998716	0.422724	0.72104	0.418855	0.999405

Figure 2-1 Typical numerical microarray data

In either approach, computational methods were applied to expression data to extract the most important genes or cluster of genes. Biological knowledge was only used to find a meaning for the output. There were several problems with this approach. First of all, it is difficult and time consuming for biologists to look at each gene or cluster of genes to find their functionality in the molecular biology context. This is because a single gene could have several functions in the body and it is not known which functionality is related to the current situation. In addition, biologists are interested in the interactions between genes, and that approach does not consider any dependencies between genes and does not tell us anything about gene relationships. Furthermore, this approach is not reliable as it may detect effects rather than causes. There are some factors which will change considerably due to a general malfunction in the body, called downstream

effectors (Chuang, Lee et al. 2007). Computational methods which consider only the amount of gene expression identify the gene with the highest changes in that disease, however, the same genes could turn up in many other diseases. Finally, this approach is very sensitive to noise because it does not deal with the function of each gene. Therefore, it is more likely to process any kind of noise or wrong value, which can lead to an incorrect result that does not make biological sense.

In 2005, a group of researchers (Subramanian, Tamayo et al. 2005) reviewed three different studies related to lung cancer by three different groups in Boston, Michigan and Stanford (Bhattacharjee, Richards et al. 2001; Garber, Troyanskaya et al. 2001; Beer, Kardia et al. 2002). They revealed that those three different studies only have one gene in common among the top 100 genes which are reported to be correlated with poor outcome for lung cancer patients. This study also found that no genes in either study were strongly correlated with the result at a significance level of 5% after correcting for multiple hypothesis testing.

There are a number of reasons for the lack of agreement between different studies. The first reason is using different methodologies such as different pre-processing steps, different normalization and transformation procedure, different modelling and, finally, different gene selection methods and algorithms. Another reason for the lack of agreement is related to the data having different samples, choice of patients or poor experimental designs such as noise. Lastly, the influence of underlying biology such as the presence of tumour subtypes and an unsuitable model or experiment for considering the important factors can make a huge difference between studies (Wilson and Pittelkow 2007).

One of the main causes of not getting a good result by conventional methods is that those methods consider each gene separately without considering its dependency on other genes. The fact is, genes do not operate alone in the cell; they operate in a sophisticated network of interactions that researchers have only recently started to investigate (Al-Shahrour, Minguez et al. 2006). Therefore a single gene analysis approach does not match with the nature of the problem. For example, a 20% increase in the expression

level of a group of genes in the same pathway[1] may be more important than a 20-fold increase in a single gene (Subramanian, Tamayo et al. 2005). Looking at a single gene usually leads to a non-unified biological result.

Another key reason for the above-mentioned problems is that they do not use any information from the domain to extract a plausible pattern from the data. In the presence of noise, redundancy and variability, using domain knowledge can help prevent a false result. The researchers who compared lung cancer studies suggested a different approach to analysis of microarray data, which they called *Gene Set Enrichment Analysis (GSEA)* (Subramanian, Tamayo et al. 2005). Their method proposed a different perspective on the problem. The method starts with some gene sets known to be involved in a biological function, such as cell death (apoptosis), and determines whether they can identify a difference between a normal versus diseased group. Those gene sets which significantly differ between two groups are chosen as the most informative gene subsets. They utilize existing knowledge about biological gene pathways to get a result which is biologically meaningful. Their statistical method looks at the most coherent and modest changes in a group of genes instead of a dramatic change in the individual genes. Using this approach, they detected a considerable number of genes in common in different studies related to lung cancer and diabetes that were previously not identified by earlier approaches. This study has led to a new generation of tools for microarray analysis which use biological knowledge from the outset to try to find biologically meaningful changes in the data (Al-Shahrour, Díaz-Uriarte et al. 2004; Al-Shahrour, Díaz-Uriarte et al. 2005; Lee, Braynen et al. 2005).Subsequent studies looked for more information from the domain knowledge such as protein interaction networks or transcription factors to define group of genes which are related functionally. Another group aimed to develop a statistical function to measure the changes more accurately for small size datasets when the number of samples is less than ten. This condition occurs frequently in the research laboratories where there

[1] A pathway is a series of chemical reactions occurring within a cell between genes, proteins, and other elements to perform a function.

are a few samples and a couple of hundred genes. ROAST (Wu, Vaillant et al. 2010), developed as a function inside the Limma package (Smyth 2005), has the answer for such a situations which GSEA (Gene Set Enrichment Analysis) cannot handle properly. A review of the methods for gene set analysis can be found in a recent survey (Abatangelo, Maglietta et al. 2009).

In recent decades, by developing a System Biology approach, microarray experiments have been used for the systematic study of gene interactions and Gene Regulatory Networks. In the next sections we will describe the principal of GRNs and review the computational methods for GRN discovery from microarray data.

2.3 Gene Regulatory Network (GRN)

A Gene Regulatory Network is defined as a graph that represents groups of genes that are activated or inhibited and their interactions regulate certain biological functions, such as metabolism and development. They are dynamic objects that continuously sense the environment and coordinate their operation accordingly. The core of GRN operation is based on gene expression.

The expression of one gene can be controlled by the specific quantity of a target protein which is a product of another gene. These proteins are called transcription factors. In general, proteins are key players inside cells. Each gene produces a unique protein/s. These proteins participate in many cell functions and also those proteins, which are transcription factors, play a key role in the regulation of gene expression. Genes are linked to each other through these proteins. The gene regulation (expression) system consists of three major elements: genes, cis-elements and regulators. Cis-elements are a region of genes that a regulator binds to and controls the expression level of the gene. Cis-elements, also known as promoters, generally consist of a few 100bp upstream of the Transcription Start Site [2] of the gene. Regulators are often those proteins called

[2] Transcription Start Site (TSS) is one base pair and is where the transcription of a gene to RNA begins

Transcription Factors (TFs) (Filkov 2006). Transcription Factors (TFs) are a protein binding domain which can recognize the sequence of the cis-element of a particular gene or genes and binds to it. For transcription to begin, the TFs bind to cis-elements along with RNA polymerase II and other proteins and make a Transcription Initiation Complex (TIC). RNA polymerase II is a molecule which has the ability to read the DNA sequence and make a copy of the code. Once TIC is formed, the transcription begins and the RNA polymerase II moves downstream, producing RNA until a stop signal in the DNA is reached. The stop signals are particular codes on the DNA sequence which can be recognized as stop signs. This process produces mRNA molecules which contain a copy of the gene's code. These mRNA molecules later on translate into proteins to perform functions such as acting as an enzyme in a metabolic process. Some of these proteins function as TFs to regulate other genes.

In general, TFs can control the transcription of a gene in two ways: one is through binding to cis-elements of the gene and making TIC complex, as described above, the other way is by directly binding to the upstream or downstream of the Transcription Start Site of the gene (Kleinjan and van Heyningen 2005).

Transcription Factors (TFs) can be in two states and transit between the active and inactive states. After becoming active, they bind to the regulatory regions of genes and change the level of expression of these genes. At the qualitative level, a transcription factor can activate (positive effect) or inhibit (negative effect) a gene target. A dual effect is also sometimes observed, which may be either positive or negative, according to the circumstances (Dardel and Kepes 2006).

Regulatory genes which produce TFs may themselves be regulated, and target genes may themselves be regulatory, in which case they participate in a genetic regulatory pathway or cascade. If such a regulatory pathway is closed onto itself, it forms a feedback circuit. Some genes are self-regulatory which form a unary feedback circuit (Kepes 2007). Considering the number of genes and all the possible interactions and regulations between them depending on different conditions and stages of the cell, the system is so complex which makes it hard to elucidate.

Much of the pioneering research in understanding the mechanisms of transcriptional regulation has been carried out on model organisms such as yeast. It is only recently that the technologies have become available to perform in-depth analysis of regulation of transcription in humans, which is inherently complex. In higher eukaryotes such as humans, the number of TFs is far less than the number of genes that need to be regulated; therefore, transcription factors act in a cooperative, orchestrated manner to initiate the transcription of a particular gene (Maston, Evans et al. 2006). Knowing the correct combination of transcription factors controlling a gene's expression is crucial to determining the correct function of any given gene; however, understanding transcriptional regulation in humans is a difficult task. One of several reasons for this complexity is that only one part of gene regulation occurs through the regulation mechanism. The other part happens through different mechanisms such as post regulation mechanisms including non-coding mRNA3, post-translational modification of proteins, RNA processing and transport (in eukaryotes). This means that it is not possible to extract all the information necessary to build a gene network solely from mRNA microarray.

The genes, regulators and the regulatory connections between them form a schema called a gene network. A directed graph, in which the nodes are genes and edges represent the control relationships between them, can be used to model these networks. A GRN is like a skeleton that provides a qualitative framework on which quantitative data can further be superimposed for reasons of quantitative modelling and simulation (Potapov 2008). Depending on the degree of abstraction, there are different levels of modelling of gene networks.

The classic way of building a gene network is to extract information from individual studies about individual links and then merge them together. There are some known networks available in KEGG (2009) and other databases publicly available that are

[3] Non-coding mRNA is a functional mRNA molecule that is not translated into proteins. They perform some important functions inside a cell.

extracted in this way. This classic way of extracting networks is really time consuming and expensive, and with the emergence of new large scale technologies there is a need for automating this process.

Some of the already known networks give us useful information. The complete gene transcription network of single cells such as E. coli and S.cerevisiae are now known. Figure 2-2 shows a typical GRN built by a computational method. We can see in this figure that the structure of GRNs is not similar to a random network. The biological networks in general and GRN in particular exhibit a certain property called scale-free. In a scale-free network when there is an increase in the number of nodes the number of edges does not scale up with power of two as we see in the normal network. Therefore their structure follows a certain shape. Most genes are engaged in only a few interactions, but a few genes are linked to a significantly higher number of other genes. These highly connected genes function as hubs. In Figure 2-2 some of the hubs are distinguished by a circle around them and it can be seen that they have more than five other nodes connected to them.

The scale-free topology provides robustness against random failures. These networks are also compact and display increased clustering. Some of this structural knowledge can be used to model gene networks effectively. In Section 9.2 we will review these properties in details and we use them in our second and third approach to build an effective algorithm.

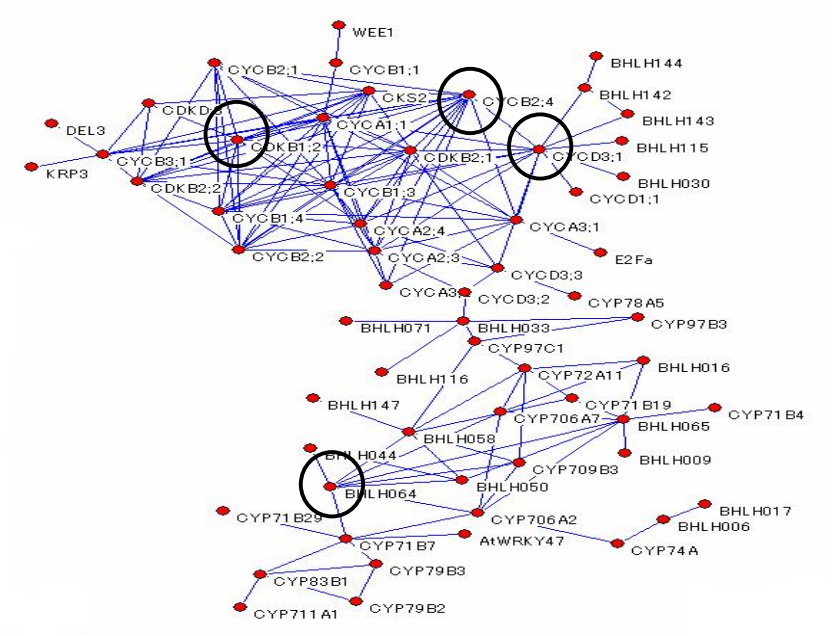


Figure 2-2 An example of a Gene Regulatory Network (A part of the cell cycle based on http://prime.psc.riken.jp) (Kenji Akiyama, Eisuke Chikayama et al. 2008)

There are different levels of abstractions and details for the representation of GRNs. In general, there are two categories: continuous and discrete modelling. A simple and common approach to discrete modelling is a Boolean model, in which each relationship is labelled as 0 or 1 (that is, each pair of genes either is related or is not related). The simplicity of this approach makes it much faster to calculate a large scale network, but doesn't provide any quantification of the strength of relationships. In contrast, in continuous modelling the edges have a real value indicator of the strength of the relationship. The graph can also be unidirectional or directional. In Figure 2-2 the graph is not directional and has a Boolean representation, therefore there is not any label to represent the strength of the relationship. This way is the most abstract representation of GRNs. In the next section, we will describe computational methods for reverse engineering of GRNs.

2.4 Gene Regulatory Network (GRN) Discovery

The study of gene regulatory networks has received major attention recently due to the development of experimental techniques like microarrays. There are different ways of inference of Gene Regulatory Network ranging from computational methods to methods from laboratory experiments and building them by hand.

In general there are several molecular biology techniques for building GRNs: Genome-based methods, High-Quality reconstruction with literature survey of interactions and finally reconstruction of the Transcriptional Regulatory Networks (TRN) from curate regulatory interactions or knockout experiments. In the Genome-based methods, sequence and location information are used in order to find relationship between genes and reconstruct GRN. Another way for reconstruction of GRNs is mining related literature and extracting information about any possible connection of a single gene with others. Usually, a single paper gives us information about the relationship of a gene with another gene or sometimes multiple genes. By accumulating this information, one can build a GRN related to a specific situation or condition, which has the best accuracy compared with other methods for GRN discovery. Finally, the last method tries to build a GRN by perturbation or knockout of genes in laboratory experiments. The technology for this is usually microarray. This is the most widely used method for reconstruction of GRNs (Feist, Herrgård et al. 2009). In this method the data consists of measurements at a steady state, following multiple perturbations such as gene over-expression, knockdown, drug treatment or at multiple time points following one perturbation (time series data).

Obviously, laboratory experiments, such as genome-based or curate and knock out experiments, combined with extracting information from the related literature, is a more accurate way of building GRNs and results in a better quality GRN. However, such a manual GRN construction by an expert is really time consuming and is suitable only for a small-sized network with few genes. The number of parameters and possible combination of the genes scale up drastically with an increasing number of genes. This is a combinatorial problem in nature. There are also many sources of information and

domain knowledge that cannot be considered all at once by a human. For mining such a huge amount of related literature, computer science provides a solution. The text mining technology helps to mine the molecular biology literature in an automatic fashion (Muller, Kenny et al. 2004). This is much more applicable than manually building GRNs. In all of the above-mentioned techniques, we need the assistance of computational methods to analyse and process the data. Computational modelling and algorithms is an essential part of GRN discovery due to the large amounts of data to be considered and the combinatorial nature of the problem.

In this thesis, our focus is on the building of GRNs using a genome based method which are the most widely used methods for GRN discovery. We used a steady state data related to multiple samples in the same condition to discover the related GRN.

Since the 1980s, in two sequential decades, a variety of mathematical formalisms for describing GRNs has been proposed. The result of these efforts is a wide range of different models with different considerations and assumptions. Currently, major formalisms for modelling GRNs are directed graphs, Bayesian Networks, Boolean Networks and their generalizations, ordinary and partial differential equations, qualitative differential equation, stochastic master equations and rule-based formalisms.

Each of these models considers different assumptions. Some models take into account the inherent stochastic nature of chemical reactions. On the other hand there is deterministic approach which considers a deterministic function and relation between genes. Simple stochastic models can be used to induce how gene expression noise can be manipulated experimentally to improve the network functions or change its function. Stochastic modelling is one step further than deterministic and deals with details of the process and temporal aspects of the interaction and therefore has more compatibility with the real world. Some stochastic approaches are Gillespie stochastic simulation, Langevin modelling and Fokker-Planck modelling (Sjöberg, Lötstedt et al. 2007) which belong to the family of stochastic master equations. In addition, for stochastic modelling of more complex networks phenomenological and mass action kinetic have been used (Kaern, Blake et al. 2003; Dilão and Muraro 2010). Stochastic modelling is more compatible

with the nature of the problem; however, the effect of the probabilistic nature of individual reaction events becomes harder to estimate, when the number of elements increases. Thus it becomes unrealistic for large scale networks (Kaern, Blake et al. 2003).

This deterministic approach is based on the implicit assumption that the underlying dynamical process is at equilibrium, and that no circuits exist in the GRN. The deterministic model used to guide the construction of toggle switch (Gardner, Cantor et al. 2000) is an example of a top-down, phenomenological, deterministic approach to the modelling of GRNs. The focus in deterministic modelling is on the structure identification problem. They ignore the temporal aspects and search for causality chains among the variables at hand. The update of the network states in this model is synchronized whereas in reality gene networks are asynchronous. Examples of the deterministic approach are deterministic differential equations and Boolean Networks.

Another consideration for modelling GRNs is how to quantify the relationships between genes. There are two options, Boolean and continuous. In Boolean the relationship between genes is either 0 or 1. In continuous modelling each relationship is labelled with a number indicating the strength of the relationship. In this thesis all the proposed approaches use continuous modelling. Boolean Networks are the example of such a model. Continuous modelling offers more precise modelling by providing a flexibility to assign strength or a probability to a relationship and opens an opportunity for further information processing such as the post-processing. An example of the continuous approach is Bayesian Networks.

Here, we first will describe the major formalisms for GRN discovery then we look at their applications within the GRN discovery literature. We will mention their strengths and weaknesses.

2.4.1 Directed Graphs

The most straightforward way to model a gene regulatory network is a directed graph in which nodes are genes and edges are regulatory relationships. The vertices and edges

could be labelled for example a tuple $\langle i, j, s \rangle$ with s equal to + or -, so it can be indicated whether i is activated or inhibited by j. Figure 2-3 shows an example of such a representation such as where i=1 and j=2 the relationship is – (inhibition) and when i=2 and j=1 is + (activation).

There are many methods which try to build the network by using pairwise correlations (associations) between genes. An example of them is Relevance Network. This method was proposed based on pairwise association scores and it is a simple approach to reverse engineering of GRNs (Butte and Kohane 2000). Another example is *BioLayout Express* (Theocharidis, Dongen et al. 2009).

In this PhD thesis, we will follow this modelling in the third approach where we will use our proposed co-regulation measure to build the graph of pairwise associations between genes.

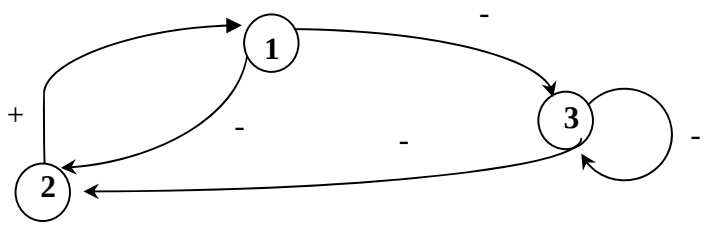


Figure 2-3 Directed graph representing a genetic regulatory network

There is also another group of studies in this formalism which try to build the network by using clustering algorithms. Several techniques for clustering time-series expression data have been proposed in the literature, based on measures like Euclidian distance, Mutual Information, linear correlation, and rank correlation (Chen, Filkov et al. 1999; D'haeseleer 2005; Do and Choi 2007). The assumption behind the clustering algorithm is that two genes exhibiting similar expression patterns over the time may regulate each other or be in a regulatory connection via other co-expressed genes in the graph. In a recent trend, measures from information theory approach have been used in this area

successfully (Margolin, Nemenman et al. 2006; Meyer, Kontos et al. 2007; Meyer, Lafitte et al. 2008).

2.4.2 Boolean Networks

The assumption behind this modelling is that the state of a gene can be described by a Boolean variable expression that is either active (1) or inactive (0) and hence its products are present or absent. In addition, the interaction between genes can be represented by Boolean functions and it is assumed that transition between states occur synchronously. Modelling GRNs by Boolean Network started with a study by Kauffman (1969b). The structure of a Boolean Network can be recast in the form of a wiring diagram which is a convenient representation for computing translations between states, as shown in Figure 2-4.

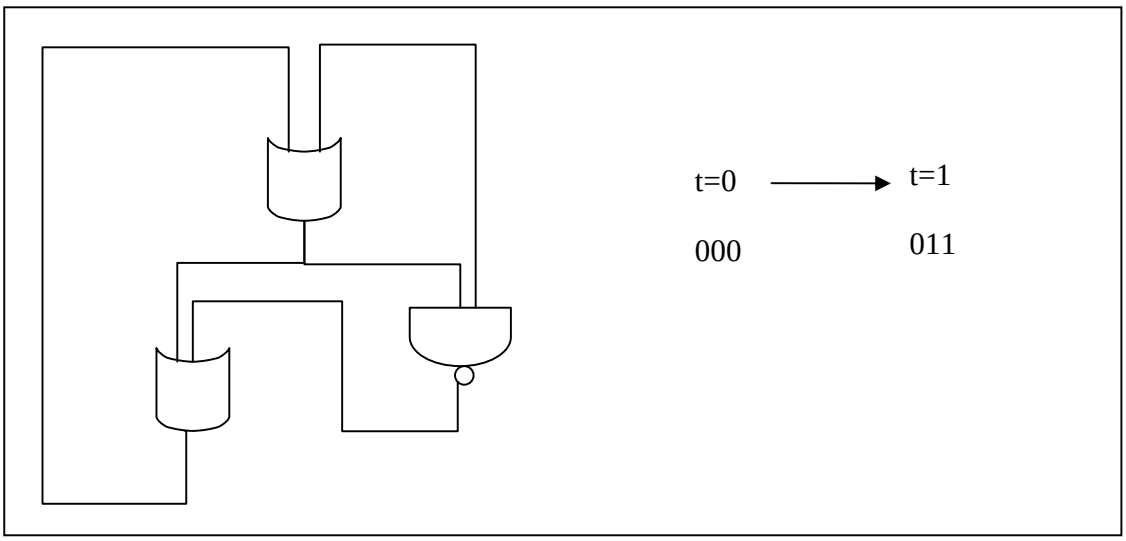


Figure 2-4 Wiring diagram of the Boolean Network

For instance, if all genes are set to inactive in t=0 then in t=1 the second and the third genes become active. Boolean Networks were among the first formalisms for which models were proposed with REVEAL algorithm developed by (Liang, Fuhrman et al. 1998) being an example. The simple modelling nature of Boolean Networks makes them suitable in the study of global properties of large scale networks (Kaufman 1979; De

Jong 2002). This property of Boolean Networks makes them suitable tools for capturing essential properties of gene-regulatory network. Thus, they are used to build an efficient tool for reconstructing networks from time series, generating random networks, performing robustness analysis via perturbation (Müssel, Hopfensitz et al. 2010) and analyse the network though signal processing methods (Xiao 2009). Recent reviews on the use of Boolean Networks can be found in Xiao (2009) and Hickman and Hodgman (2010).

2.4.3 Differential Equations

Differential equations (DE) are the starting point for quantitative modelling of complex systems and can be used to describe non-linear and dynamic systems. Ordinary differential equations (ODE) are the most widespread formalism to model dynamical systems and have been widely used to analyse GRNs. Their use goes back to the 'operon' model of Jacob and Monod (1961) and early work by Goodwin (1963).

The ODE models the expression of genes by time-dependent variables with non-negative real values. Interaction between genes is formulated as a function of other genes and differential relations as shown in Equation (2.1).

$$\frac{dx_i}{dt} = f_i\,(x) \quad 1 < i \leq n \tag{2.1}$$

ODEs were first used in the context of modelling metabolic process (Cornish-Bowden 1995) but kinetic model of a simple gene regulation process going back to the seminal work of Goodwin (1963; 1965). Calculation of a system of differential equations is complicated when the number of variables exceed more than five; therefore, a number of ways have been proposed for simplification and overcoming the scalability problem. One way is numerical simulation, which considers consecutive discrete time points instead of continuous time (Reinitz and Vaisnys 1990; Lambert 1991).There is another simplified version of ODEs called piecewise-linear differential equation which tries to discretize

values between pieces and considers a linear model (De Jong 2002; Geberta, Raddea et al. 2007).

The use of differential equations is still limited in size. In addition, verification of these methods is limited to small-sized networks, as the kinetic parameters are available for only a handful of small networks and for larger models there is not such information available.

In general, differential equations as a modelling framework have two major drawbacks. Each equation of the model requires the knowledge of one or several parameter values (such as thermodynamic constants and rate constants). At the moment it is impossible to obtain such information based on the current techniques. Thus it is difficult to build an instance of the model for large networks directly, and reverse-engineering techniques are limited in how much information they can extract from limited datasets. Moreover, deriving meaningful dynamical properties of a large differential equations system is a challenge (De Jong 2002). The advantage of using differential equations is that they can provide us with both a directed and undirected network. Their performance is also the best compared with other methods in presence of small perturbation and temporal data.

It is obvious in this modelling that we have burdened computational overhead to compute continuous parameters of the model. A combination of Boolean and continuous modelling has been used and the result is known as hybrid modelling (Schlitt and Brazma 2007). In the hybrid model, Boolean logic is used to model biochemical processes with a sharp threshold, and continuous dynamics to model processes with a slower threshold (Smolen, Baxter et al. 2006).

In the first approach of this thesis, we used a differential equations modelling. In this approach, we developed a new technique to more effectively find a system of differential equations used for modelling a GRN.

2.4.4 Stochastic Master Equations

Differential equations assume that concentrations of substances vary continuously and deterministically. Instead of taking a deterministic and continuous approach, some authors have proposed to use discrete and stochastic models of gene regulation (Gillespie 1977; Nicolis and Prigogine 1977). Stochastic master equations are a phenomenological set of first-order differential equations describing the time evolution of the probability of a system to occupy each one of a discrete set of states. Discrete amounts x of genes are taken as state variables and a joint probability distribution is introduced to express the probability that at time t the cell contains x_1 amount of the first gene and x_2 as the amount of the second gene etc.

They consider stochastic relationships between elements and model them by use of rate equations. Some of the methods in this category are Gillespie stochastic simulation, Langevin modelling and Fokker-Planck modelling (Sjöberg, Lötstedt et al. 2007).

These types of equations are more difficult to solve than deterministic equations. A simulation developed by Gillespie (1977) proposed to solve them.

2.4.5 Gaussian Graphical Models

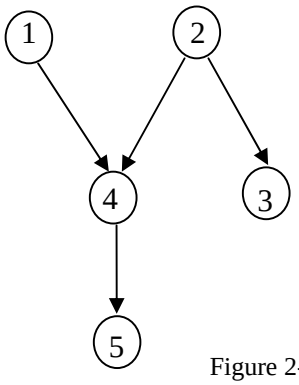
This model is based on the assumption that the expression data follows a multivariate Gaussian distribution. The graphical interaction model for the multivariate normal distribution is called a Gaussian Graphical Model and was first introduced by Dempster (Dempster 1972).

To avoid the shortcoming assumption of building a directed graph based on Pearson's correlation, this uses partial correlation (Grzegorczyk, Husmeier et al. 2008). It considers partial correlation between any two variables conditional on all the other nodes in the network. The disadvantage of this method is that the computation over the matrices requires that the number of observations exceeds the number of nodes. Considering the fact that in this application there is usually much less number of records compared with hundreds of variables, makes this requirement a problem. To overcome this problem

some methods have been proposed in the literature; an example is the work by Schafer and Strimmer (2005) which proposes a shrink estimator of covariance matrix.

2.4.6 Bayesian Networks

In this formalism, the structure of GRN is modelled by a directed acyclic graph. The vertices represent genes which correspond to random variables x_i. For each x_i a conditional distribution $P(x_i|\text{Parent}(x_i))$ is defined, where Parent (x_i) denotes the variables (genes) with direct regulators to x_i (Friedman, Linial et al. 2000).



$$P(x) = P(\boldsymbol{x_5}|\boldsymbol{x_4})P(\boldsymbol{x_4}|\boldsymbol{x_1},\boldsymbol{x_2})P\ (\boldsymbol{x_3}|\boldsymbol{x_2})\ P(\boldsymbol{x_2})\ P(\boldsymbol{x_1})$$

Figure 2-5 An example of GRN modelled by a Bayesian Network

It is a network of variables that shows the probabilistic relationship between them in terms of conditional independency relationship. For building a GRN based on Bayesian Networks there are two steps to follow, first to find the structure of the network and then to compute the conditional dependency of edges in the network. The hardest part is to learn the structure of the graph. Different methods are used to learn the structure, including Markov chain Monte Carlo (MCMC) (Friedman and Koller 2003).

Bayesian Networks are among the most used techniques in the GRN inference literature. There are two reasons for this. Firstly, they can provide models of causal influence and secondly, their probabilistic nature is well suited to the noisy nature of expression data (Friedman, Linial et al. 2000). Some of the most important studies related to GRN discovery were developed based on Bayesian Network. Pioneer of using Bayesian

Networks are Friedman and his colleagues (2000). The dynamic version of Bayesian network used for quantitative modelling of GRNs (Nachman, Regev et al. 2004). Hartmink and his team have developed a popular tool based on Bayesian Network called Banjo (Yu, Smith et al. 2004).

A famous study based on Bayesian Network is called *Module Network* (Segal, Shapira et al. 2003). In that study Bayesian Network were used along gene modules. Modules were functional gene sets which were extracted from domain knowledge. Modules were used to divide the search space at the beginning and then Bayesian Network was applied to find their dependency and relationships, and at the end these networks were combined to see the overall picture. The idea of dividing the search spaces overcomes a limitation of Bayesian Network. Bayesian Network is not an applicable tool for big datasets with thousands of features therefore dividing the search space by using modules helps to scale up for bigger problems. There is another study based on the Bayesian Network which employs domain knowledge and it considers gene sequence data in addition to known and predicted transcription factors to achieve a higher accuracy (Noto and Craven 2005).

Despite its powerful ability, Bayesian network has two weaknesses in this application. One is that it is not applicable for large networks and another is that it cannot present loops and feedback loops (He, Balling et al. 2009) Feedback loops are known to play key roles in causing dynamic behaviour of the network and are a common feature of them (Thomas, Thieffry et al. 1995). An extension of Bayesian networks called dynamic Bayesian networks are able to present loops but it has the scalability issue.

2.5 Summary

This chapter covered the background information and some of the literature related to this thesis topic. We first provided a background on basic concepts in molecular biology and then described microarray technology and its usage and a sample output. We also reviewed the challenges for the analysis of microarray data and we mentioned that these challenge are mainly due to the noise and data variability.

In the second part, we moved to the biological background of GRNs. We first described the gene regulation mechanism and process in order to understand Gene Regulatory Networks and then we described the different GRN representation and modelling. At the end, we reviewed the computational modelling for gene regulatory network discovery. We highlighted each category's strengths and weaknesses.

Gene Regulatory Network discovery literature was reviewed to provide some background on the research problems tackled in this thesis. The literature provided in this chapter is only a general overview and more detailed literature will be provided later on where we describe each of our approaches.

In the next chapter, we are going to talk about experimental setup. Firstly, we will provide a review of the literature about synthetic GRN generators and simulators then we will describe the simulator which we chose and the datasets that we produced for our experiments using that simulator. Secondly, we will explain the related literature to evaluation to provide a background for the reader to understand our evaluation metric which we will describe at the end.

Chapter 3

Experimental Setup

"Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful."

— George Edward Pelham Box

3.1 Introduction

The purpose of this chapter is to introduce the experimental data which we used in order to test our algorithms. In the first part, we will introduce the concept of simulators and why we need them to generate benchmarks for testing GRN reverse engineering algorithms. Then we will describe the literature related to simulators. Later, we will explain why we chose a particular simulator (SynTReN) for this study. We will also inform the reader about its characteristics and functionality.

In the second part, we will review the literature related to evolution and performance measures for GRN discovery in order to choose our performance evaluators. We will describe different perspectives that we can choose to evaluate output networks. We will also describe existing metrics that have been used in each perspective. Then we will focus on studies applied to the data produced by SynTReN. We will describe how they performed and what metrics were used to measure the performance.

Finally, based on those literature reviews, we will present our experimental setup. Firstly we will describe our considerations for generating different benchmarks and how our six benchmarks were generated using SynTReN. The other configuration setup will then be mentioned, such as software, tools and programming languages. Finally, we will inform the reader about what metrics we use to measure our performance and also studies with which we will compare our work.

3.2 Synthetic GRN Simulators: Introduction and Motivation

The development of algorithms to infer gene regulatory networks based on microarray expression data is a complicated task. The methods developed so far still do not provide an acceptable result (Marbach, Mattiussi et al. 2009). Also, most of the studies focus on the prokaryotic cells and still have far to go to be able to perform a discovery for advanced cells (eukaryotes) such as human cells. According to Haynes and Brent (2009) even small changes in the accuracy of datasets, or the design of experiments can make a big difference in the performance of the algorithms applied to it. Thus it is important to have a benchmark that is not dependent on any experimental error to test different algorithms. We also need test beds which are not only comprehensive enough but also realistic.

A few biologically inspired (small-size) benchmark problems have been proposed, Raf pathway (Werhli, Grzegorczyk et al. 2006), a yeast model (Cantone 2009; Cantone, Marucci et al. 2009), and finally yeast and E. coli model (Gama-Castro, Jimenez-Jacinto et al. 2008). However, experimental datasets of the appropriate size and design are still not available. There is not any real known large-scale (genome-wide) model of a gene regulatory network to evaluate and compare different methods for reverse engineering (Heckera, Lambecka et al. 2008). In such a situation, the validation strategies applied to experimentally obtained data are often limited to confirming previously known interactions in the reconstructed network. However, with such an approach, false positive interactions are not penalized. Moreover, algorithms can only be applied to data from a single network which creates bias towards that dataset. Each experiment and dataset has a bias and variation; therefore we need a collection of real datasets to have a comprehensive test bed, which is not available yet. Most of the already known networks are created from a collection of experiments found in the literature; therefore there is usually no single corresponding dataset available. We need the ability to test the performance of different algorithms in the presence of different conditions, like different

types and levels of noise, different regulatory edges and different network sizes. These are the reasons behind the need to generate well-characterized synthetic datasets that allow thorough testing of learning algorithms in a fast and reproducible manner without any bias to a particular condition.

A gene regulatory network simulator is a program that can generate similar data to microarray data as well as generating the underlying network. In this network, the nodes represent the genes and the edges are the regulatory interactions between the genes. Also, there are other types of microarray simulators which only produce the microarray data for other purposes such as classification and clustering of diseases; therefore they do not produce the underlying network. In a synthetic GRN we have the ability to change the benchmark features and produce (i) different size networks (ii) realistic (non-linear) effects like state saturation or joint regulatory action of several genes (iii) perturbation experiments like gene knock out and (iv) different types and levels of noise.

Despite the above facts, data simulators are not popular tools among bioinformaticians and there are arguments against them, such as, they do not capture reality and any algorithms developed based on them will not be able to perform well on real data (Sauro, Harel et al. 2006). Maybe this tendency against them underlies the nature of biology, which is complicated and has many variations and as the result hard to simulate all of those complications. Also, bioinformatics comes from a biology perspective that tends to take laboratory samples.

3.3 Comparison of Simulators

There are different types of benchmarks for the purpose of testing Gene Regulatory Network reverse engineering methods. Most of these benchmarks were developed in silico (by use of computational simulation) and plenty of them are available. A-BIOCHEM (Mendes, Sha et al. 2003) is an example of in silico benchmarks. Those that are based on the computational simulation usually can simulate the large gene networks with different levels of noise. Some of them just produce a random network and others produce networks in which the structure follows some properties of biological networks.

Among most recent simulators we can name GRENDEL (Haynes and Brent 2009), AGN (Mendes, Sha et al. 2003), RENCO (Roy, Werner-Washburne et al. 2008), Netsim (Camillo, Toffolo et al. 2009), Gene NetWeaver (GNW) (Marbach, Mattiussi et al. 2009) and finally COPASI (Hoops, Sahle et al. 2006) which were used in DREAM2 (Lee, Narang et al. 2009) competition to build the test benchmarks. Artificial simulators usually model the data as systems of non-linear differential equations (Zak, Doyle et al. 2001; Mendes, Sha et al. 2003).

Synthetic data generators vary from the perspective of how much their data resembles the real data. They range from generators that only produce a network based on the pure conditional dependency, to those that generate a network and corresponding microarray data based on real known networks and interactions.

An important factor for comparison of the simulators is how closely the output resembles real biological networks. Therefore, recent trends utilize partial networks from the biological domain to build a network which is similar to the real one, and then produce the expression data based on that network. In a review that we have completed to choose a synthetic data generator, we identified SynTReN (Bulcke, Leemput et al. 2006) as the best and most well known tool for our purpose at the time. SynTReN effectively uses known network parts to build the simulated network and according to the authors, its network output is the most similar one to real networks (Bulcke, Leemput et al. 2006). Our proposal involved using biological knowledge and heuristics from the known network in order to elevate the performance; therefore we needed a simulator which could produce networks similar to the domain network as much as possible and SynTReN is the best for this purpose. In addition, SynTReN has been applied to different well-known studies related to reverse engineering of regulatory networks (Meyer, Lafitte et al. 2008). There are many details available describing the exact performance of the well-known systems using SynTReN data. This information helped us to figure out in which condition our algorithm could improve the performance of the state-of-the-art algorithms. SynTReN has been cited 66 times so far in the related literature and was also used by minet library (Meyer, Lafitte et al. 2008) in Bioconductor package (Gentleman,

Carey et al. 2004) to generate artificial GRNs for the purpose of testing different algorithms.

3.4 How does SynTReN work?

During three steps SynTReN produces expression data and the underlying network.

1. In the first step, a network topology is selected from a known network using either of two selection strategies. One selection strategy is neighbourhood addition and another is clustering addition.
2. In the second step, transition functions and their parameters are assigned to the edges in the network. To model regulatory interactions, Hill Kinetic equations and Michaelis-Menten are used. This allows the production of a variety of interaction types which are likely to occur in real biological systems, such as nearly linear or very steep interactions. All transcription rates are assumed to be in a steady-state regime. This is in contrast with other simulators which are based on coupled differential equations and as a result, they can only produce a limited sized network. Thus SynTReN, because of this simplification, can simulate large networks of thousands of genes.
3. In the third step, mRNA expression levels for the genes in the network are simulated under different conditions. After this, there is an option to add two types of noise to the data: experimental and biological noise. Finally, the data is normalized, and the scaled microarray measurements and the corresponding network are generated.

In SynTReN several parameters are user-defined, such as kinetic parameters of kinetic equations; therefore a user can generate datasets with increasing levels of difficulty in order to generate benchmark problems for their system.

Biological GRNs have specific structural properties such as the small world property and the scale-free property. They also contain specific structural motifs which do not happen in random graphs with the same in-and-out degree. Thus it is quite difficult to produce a

network similar to a real biological network. However, SynTReN is one of the best simulators in terms of the similarity of its result to real biological data.

3.5 Literature Related to Evaluation

In order to have an idea of how to compare the results of our method, we did another literature review related to scoring and evaluation. The following two sections will present the output of this review. The first section (3.5.1) will describe the possible indications of performance found in the literature and the second section talks specifically about other related studies applied on SynTReN and how their performances were evaluated.

3.5.1 Performance Scoring

Comparison of algorithms and their performance is necessary not only to order the algorithms, but also to find out in which condition a particular algorithm can perform the best. In this way we obtain not only information about different systems and algorithms, but we can also get insight into the underlying problem by understanding how we can perform well in a particular condition.

In general, the quality of inference algorithms can be evaluated based on the following criteria:

1. The ability to identify the true structure of the GRN
2. The ability to identify the behaviour of the GRN
3. The ability to identify the gene-gene interactions
4. The ability to identify the correct labels for interactions

Similarity of the structure of the target GRN with the output GRN can be measured in terms of the main indicators of the network properties such as scale-free, small world or detailed and local properties such as clustering coefficient and average path length. For comparison of results in terms of the network structure specifically having small network properties, a model fitting index was proposed (Zhang and Horvath 2005). This index is

defined as the coefficient of determination (*R2*) of the linear model constructed by regressing $\log(p(k))$ on $\log(k)$, where *k* represents the degree of a given node (the number of edges connecting to the given node), and *p(k)* is the frequency distribution of the degree *k* in the co-expression network. Also, the similarity of the networks' structure can be measured in terms of the local structure and network motifs. An example of that is considering the rate of successful prediction of feed forward loops (Marbach, Prillc et al. 2010).

The second criterion, the ability to identify behaviour of the GRN, is only applicable when there is a quantitative modelling approach like differential equations which considers time and state of the network.

The third criterion, the ability to identify gene-gene interactions, is the one most used in the literature. It is a straight forward and simple way to measure an algorithm's performance and is frequently used in information retrieval and statistical inference. The quality of two networks can be measured based on only the detected interactions (third criterion) or more precisely based on the exact nature of these interactions such as activation ("ac"), repression ("re") (fourth criterion).

We have used the third and the fourth criterion, the ability to identify the correct labels of the interactions, to compare the performance of our work with other well known works in this area. The reasons behind this decision will be discussed at the end of this section.

For measuring the third characteristic the following criteria from information retrieval have been used in the literature.

TP (True Positive) = the number of correctly inferred edges;

FP (False Positives) = the number of inferred edges that are incorrect;

TN (True Negatives) = the number of missing edges in the inferred network that are also missing in the true network;

FN (False Negatives) = the number of missing edges in the inferred network that are an edge in the true network.

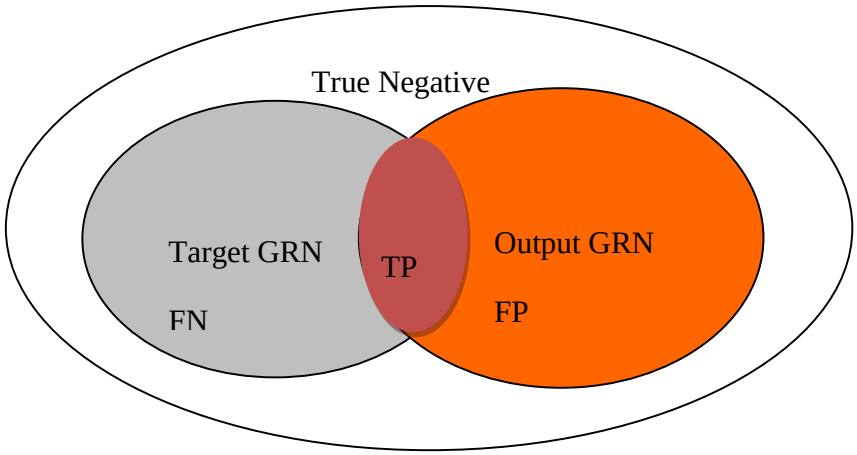


Figure 3-1 Measures of accuracy and specificity illustrated in the form of a Venn diagram

Based on the above definition some performance scores are defined as follows:

$$Sensitivity = recall = \frac{TP}{P + FN}$$

$$Specificity = \frac{TN}{TN + FP}$$

$$Precision = \frac{TP}{TP + FP}$$

Each of those scores, recall or precision, is hardly informative alone and usually combinations of these measures are used to show performance of a system. The main ones are defined as follows:

$$J = Jaccard\ index = \frac{TP}{FP + FN + TP}$$

$$F\_measure = \frac{2 \times precision \times recall}{precision + recall}$$

Based on the above metrics, systems can adjust their parameters to obtain networks with different characteristics. ROC (Receiver Operating Characteristic) curve is a common visualization tool that presents false positive rates versus true positive rates. In gene regulatory network discovery the situation is that there is usually a high number of false

positives and ROC is of limited use under such a condition. Thus instead of ROC, precision versus recall (PVsR) which is more informative has been used just to compare true edges versus inferred edges. PVsR used in many studies for measuring performance of algorithm such as in ARACNE (Margolin, Nemenman et al. 2006). Figure 3-2 presents an example ROC curve.

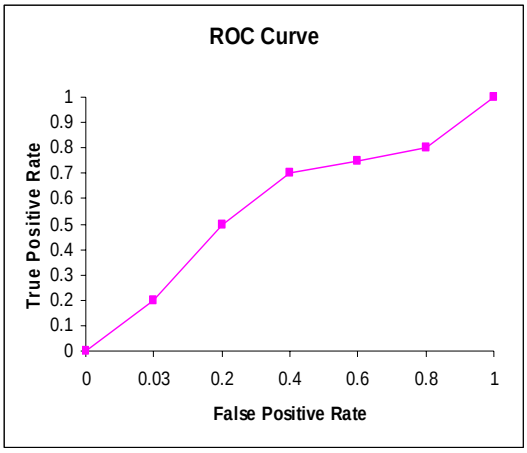


Figure 3-2 An example of a ROC curve

In the PVsR diagram the horizontal axis presents recall and the vertical axis presents precision as shown in Figure 3-3.

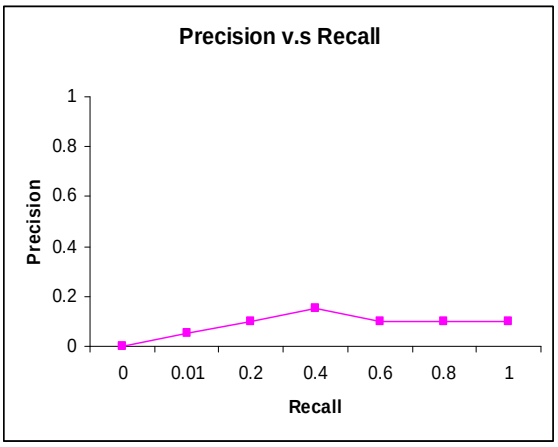


Figure 3-3 An example of a PvsR curve

The area under the curve (AUC) is another indicator of performance of systems which has been used for gene regulatory network inference (Werhli, Grzegorczyk et al. 2006). An example of using AUC is in the DREAM competition (Madar, Greenfield et al. 2010). AUC is the rate of the prediction value for a randomly picked link being larger than that for a randomly picked non-link in the test set. According to Soranzo et al. (2007), an AUC close to 0.5 corresponds to a random forecast, AUC< 0.7 is considered poor and AUC<0.8 is considered good. Usually for the purpose of gene regulatory network inference AUC is about 0.6 (Soranzo, Bianconi et al. 2007).

3.5.2 Studies applied on SynTReN

In the literature related to GRN inference, the performance of systems was usually tested in both artificial and real datasets. The problem is that every study used different artificial datasets and there is not common agreement in using artificial data, as they are not usually welcomed in the area.

However, there are studies that try to benchmark different algorithms on single datasets. An example is a study by Leepmut (2008) that tried to benchmark well known systems on single-source artificial data. They measured the performance of the well-known systems for GRN discovery on a range of synthetic networks produced by SynTReN (Bulcke, Leemput et al. 2006). They produced different datasets with different sizes, different amounts of noise, and different amounts of complicated interactions. Then, they compared the performance of well-known algorithms such as ARACNE (Basso, Margolin et al. 2005; Margolin, Nemenman et al. 2006), Genomica (Segal, Shapira et al. 2003) and SAMBA (Tanay, Sharan et al. 2004) under those different conditions. The authors measured the number of correct and false edges detected by the algorithms and then calculated the F-measure and Jaccard Index for each of those algorithms on each specific dataset.

Their results revealed some interesting characteristics of those systems. For example, they reported that ARACNE achieves the maximum performance on noiseless data sets, but Genomica sometimes benefits from small noises. Their results confirmed the added

value of synthetic data in revealing operational characteristics of inference algorithms. These characteristics are unlikely to be discovered by using real biological microarray data with the limited size and options. This is an example of the advantage of using a synthetic data generator which provides us with the opportunity to test the performance of algorithms under different conditions and provides us with the opportunities to rank algorithms based on how they perform in general not only in a specific dataset.

SynTReN was also used in a Bioconductor package called *minet* (Meyer, Lafitte et al. 2008) which has the implementation of known systems for GRN inference such as ARACNE (Basso, Margolin et al. 2005; Margolin, Nemenman et al. 2006) and CLR (Faith, Hayete et al. 2007). In addition *minet* has the implementation of many association measures such as Mutual Information and correlation coefficient. In *minet,* all these algorithms were tested on a dataset produced by SynTReN and their performances were measured in terms of precision and recall. The ROC curve was used to visualize these measures. Having such information about the performance of the well-known systems on a dataset produced by SynTReN added an additional weight to choose SynTReN.

A similar study has been done by Altay and Emmert-Streib (2010) with datasets produced by SynTReN along with some famous inference algorithms such as ARACNE (Margolin, Nemenman et al. 2006), CLR (Faith, Hayete et al. 2007), MRNET (Meyer, Lafitte et al. 2008) and RN (Butte and Kohane 2000). They compared the performance of the above systems from another perspective. They used local network-based measures extracted from Emmert-Streib and Dehmer (2009) to compare the performance of the systems. They measured the type of interactions extracted by each of those methods, basic motifs of three or four genes, and the number of in-degree of the nodes and so on. In this thesis, we did not use such a measure because in that research the false positive rate was not taken into account and that is an important factor to be considered. We decided to follow the most common path for performance evaluation which was also followed in the previously mentioned study by Leemput (2008). Thus, our algorithms will be evaluated based on precision-recall and F-measure.

3.6 Experimental Data

In order to test our hypothesis, we needed several platforms to test our models and algorithms in different conditions. As it was mentioned earlier, we could not find such real test beds as the number of the known networks and corresponding data for testing purposes is limited in size and variety therefore, the above facts led us to use a synthetic data generator. We used SynTReN, a synthetic data generator (Bulcke, Leemput et al. 2006) to generate benchmark networks with the similar structure to the known network of different sizes, different properties, different amounts of noise and different percentages of each type of interactions. Parameters that can be set in SynTReN are as follows:

Network Size: Large network size is always a problem. A suggestion to overcome this problem is to use domain knowledge to reduce the search space.

Graph Topology: In SynTReN there is an option to produce the network with a neighbour addition method or with cluster addition. The first one uses a random graph model. The second one uses subnetworks from domain knowledge and gradually adds nodes to that. The neighbour addition method shows more variation for the median in-degree compared to the cluster addition method. The cluster addition method produces networks similar to domain networks in terms of the network properties (Leemput, Bulcke et al. 2008). These observations also hold for topological characteristics other than average directed path length and average in degree.

Noise Type and Amount of Noise: There are several types of noise in microarray data: biological noise, experimental noise and input noise. In SynTReN biological noise is modelled in the transition function. The transition function is a nonlinear function in SynTReN. The function type is steep sigmoid. Biological noise propagates through the network, but experimental noise does not.

Amount of Expression Data: This is a challenging area for almost all data mining algorithms. The condition in which the number of samples compared to the number of genes is so low causes the *curse of dimensionality*. In such a condition it is hard to infer

any pattern from the data. The number of records directly affects the performance of any algorithm for identification of co-expression in microarray data (Yeung, Medvedovic et al. 2004). This problem becomes extremely important in practice for laboratory research experiments, where the number of experiments is usually less than 10 and the number of genes is more than a couple of hundred. This is a significant problem mentioned in the literature and important research has been done in developing efficient algorithms for analysis of microarray data in such a condition (Smyth 2005; Wu, Vaillant et al. 2010).

Interaction Types: Nonlinear interactions act as a buffer to mask the activity of downstream genes in an interaction cascade.

Based on the above mentioned characteristics, we ran SynTReN six times with different parameters and we produced six different networks and corresponding microarray data. The domain knowledge network which its subparts was used to produce these six networks was E. coli full network (Gama-Castro, Jimenez Jacinto et al. 2008).

The Table 3-1 summarizes the information regarding our six different datasets that we generated using SynTReN. These datasets were used in our experiments in this thesis.

Table 3-1 Datasets generated for benchmarking by SynTReN

Dataset	Number of Experiments	Number of Genes	Subnetwork selection method	Biological Noise	Experimental Noise	Probability of Complex interactions
1	100	200 (100 background, 100 foreground)	Neighbour addition	0.1	0.1	0.3
2	100	200 (100 background, 100 foreground)	Cluster addition	0.1	0.1	0.3
3	100	200 (100 background, 100 foreground)	Neighbour addition	0.1	0.1	0.4
4	100	200 (100 background, 100 foreground)	Neighbour addition	0	0.1	0.3
5	50	200 (100 background, 100 foreground)	Neighbour addition	0.1	0.1	0.3
6	100	200 (100 background, 100 foreground)	Neighbour addition	0	0	0.3

The first dataset was generated by the default values of SynTReN as is presented in Figure 3-4. In these default parameters, the number of experiments

was 100. The number of genes was 200. The 200 genes consisted of 100 genes with real genes' names which made the foreground network, and the other 100 genes starting with "bgr_" followed by the real genes' names, made the background network. Only the foreground genes are directly or indirectly activated/inhibited by varying the external conditions. The role of the background network is solely for generating 'background data'. Expression values of background genes vary solely due to the effect of different types of noise. The method for creating the first network was neighbourhood addition. The amount of experimental noise and biological noise was by default 0.1.

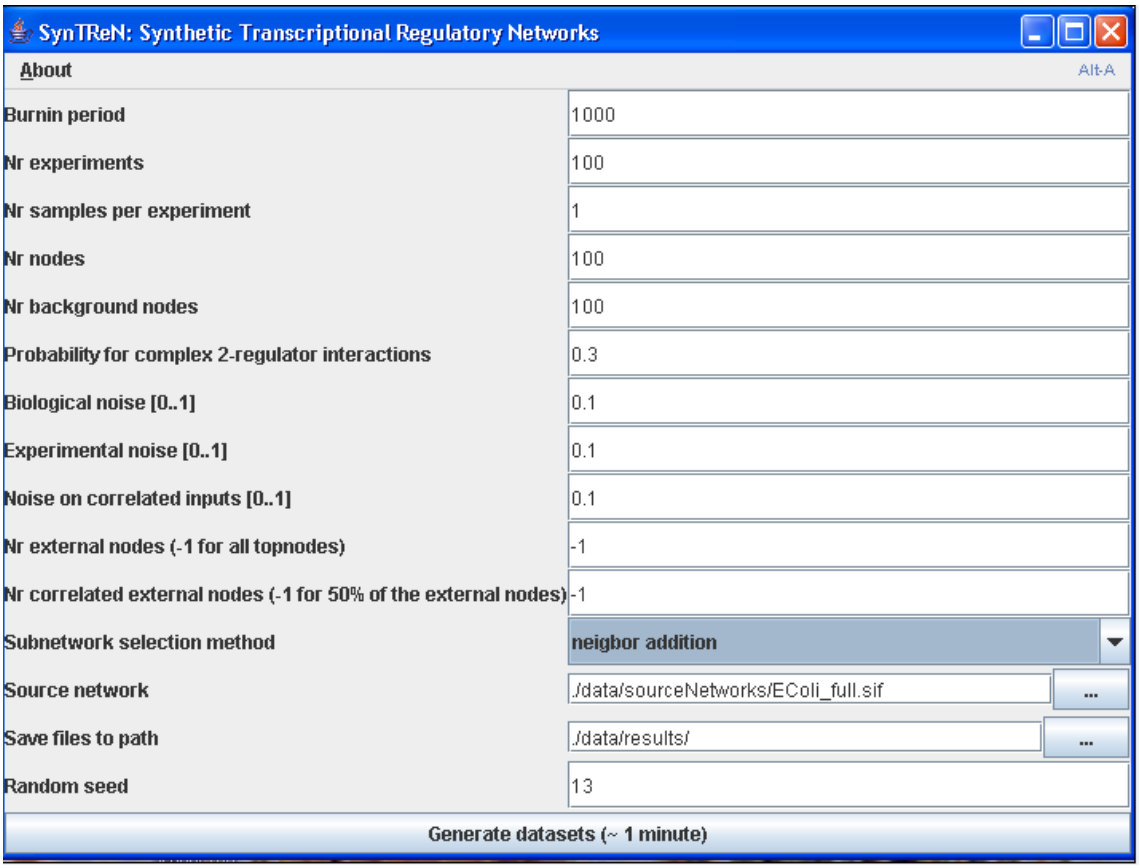


Figure 3-4 A screen shot of SynTReN generating the first dataset

The second dataset was generated by changing the subnetwork selection method to the cluster addition. The rest of the parameters were the same. By choosing the cluster addition the result network is going to be more similar to the known network.

The third dataset was generated like the first one but we only increased the probability of generating the complex 2-regulatory interactions from 0.3 to 0.4. The third dataset was generated to see the effect of increasing complicated interactions on performance.

The fourth dataset was produced based on the first dataset by removing the biological noise to see the change of performance in the absence of biological noise.

The fifth dataset was produced based on the settings of the first dataset. The only difference was the number of experiments was decreased by half to 50, to see the effect of changes when the number of experiments is low. The low number of experiments is a big challenge for any algorithms for GRN discovery.

In the sixth dataset we set the amount of biological and experimental noise to zero in order to generate a dataset free of such noise and test our algorithms' performance in this condition. These six datasets were used in the set of several experiments that we will illustrate in the following chapters where we will explain our second and third approaches.

3.7 Our Performance Measure

In the previous section we mentioned that there are several aspects based on which we can compare different algorithms and models for GRN discovery. Even for the same network we can consider several aspects to evaluate it , such as similarity of the structure of the output network with known networks of the same organism, number of true detected edges and number of false detections (false positives). Here we refer the reader to the discussion in section 3.5.2 about comparison of different algorithms applied to SynTReN data. Based on the fact that algorithms are usually compared in terms of the number of true detected edges and number of false ones using F-measures, ROC curve or URCOV (Leemput, Bulcke et al. 2008) , we used F-measure for the performance

measure. F-measure is the weighted harmonic mean of precision and recall. It is also called balanced F-score and was first introduced by van Rijsbergen (1979). Since then it has been widely used as a measure for combining precision and recall. We did not compare our result with different methods in terms of similarity of the structure of the network. The reason for that was that only some of the well known methods use the domain knowledge. In addition, it is methodologically questionable to compare the algorithms which do not use such information with the ones which do use it, like our method. It seems obvious that a method which involves information about the source network will result in a more similar network to the source network. Nonetheless, we used a visualization tool Cytoscape (Shannon, Markiel et al. 2003) to represent the structure of our result networks. Our networks quite resembles the known network in terms of the structure of the network compared to networks produced by other systems, especially those that do not use any structural properties of the known networks.

3.8 Software and Tools

All the programs and algorithms were developed using Python 2.6 in the Eclipse environment. NumPy and SiPy and StatsPy were also used for matrix and numerical operations and statistical functions. RPy was used for accessing R and Bioconductor packages inside the Python environment and we used the 9.1 version of R. We also used *minet* (Meyer, Lafitte et al. 2008) inside the Bioconductor package (Meyer, Lafitte et al. 2008). *Minet* is a library including implementation of some of the most famous algorithms for GRN discovery such as ARACNE (Margolin, Nemenman et al. 2006), CLR (Faith, Hayete et al. 2007) and *mrnet* (Meyer, Kontos et al. 2007). In addition, *minet* comes with some artificial datasets originally generated by SynTReN and modified to be used in this package for testing purposes.

For network (graph) presentation, filtering and so on, Cytoscape (Cline, Smoot et al. 2007) was used, and for generating benchmark datasets SynTReN software was used as mentioned earlier.

3.9 Summary

In this chapter, we studied the literature related to simulators for the purpose of reverse engineering of gene regulatory networks. We discussed that we need simulators because real benchmarks of different sizes and variety are not available. We also mentioned the advantages of using them, which was the ability to compare different algorithms based on different perspectives and different conditions. We mentioned some of the known simulator software, and then we discussed why we chose a particular one called SynTReN. We explained how SynTReN works in general and mentioned that SynTReN uses partial subnetworks from the domain knowledge networks to produce networks with the similar structure to the known networks.

In the second part of the chapter, we provided another literature review related to evaluation. We also provided a focused literature review on the measures used by studies applied on SynTReN data.

In the third section, we explained the experimental setup of this thesis. Firstly, we showed how we produced six different datasets using SynTReN. These six benchmarks enabled us to evaluate our algorithms' performance in different conditions, such as presence of different type and levels of noise and complicated 2-regulator interactions. These datasets will be mentioned later on, where we will describe our approaches. Secondly, we mentioned the evaluation metrics we chose for testing our algorithms according to the mentioned literature. Finally, we provided a list of software and tools which were used in this thesis.

In the next chapter, we will provide the background related to evolutionary computation techniques for a reader who is not an expert in this area, as our first and second approaches are based on evolutionary techniques. Firstly, we will explain Genetic Algorithm, Genetic Programming and Gene Expression Programming. We will then explain hybridization techniques and will move to the memetic algorithms literature.

Chapter 4

Introduction to Evolutionary Computation

"Science is built up of facts, as a house is with stones. But a collection of facts is no more a Science than a heap of stones is a house"

-H. Poincaré

4.1 Introduction

As mentioned in the Introduction chapter, we followed three approaches for Gene Regulatory Network discovery. Two of these approaches used techniques from evolutionary computation to solve the problem. The first approach used a new Gene Expression Programming algorithm to solve a model of GRN formulated by a system of differential equations. The second approach tried to find sub-networks by using a new combined genetic algorithm. Therefore, here we will present an introduction to the evolutionary computation area and review the techniques that we used in this thesis. We will also provide an introduction to hybrid methods, as both of the methods which we used in this thesis are hybrid methods.

Evolutionary algorithms originated from genetic algorithms (GA). GA was coined by John Holland in 1975 in his book 'Adaptation in Natural and Artificial systems'. Although, Holland was known as the creator of GA, he was not the first person who thought about using the principle of natural evolution in computer science.

In the 1970's in Germany, Rechenberg (1971) and Schwefel (1974) developed the idea of the Evolutionary Strategy. In the USA, Fogel, et al (1995) implemented an idea which they called Evolutionary Programming. The common concept in both ideas was using mutation and selection (Reeves 2003). Despite achieving considerable results,

Evolutionary Computing did not get enough attention until 1980, mostly because these techniques needed enormous computational power which was not available at that time.

Following Holland, his PhD students continued work on his ideas in evolutionary computation optimization and a series of further studies led to the first conference in 1985. One of Holland's graduate students, David Goldberg, produced his doctoral thesis which won an award for its application to gas pipelined optimization and, consequently he published a book 'Genetic Algorithms in search optimization, and machine learning' (Goldberg 1989). This book was the final element towards sustained development of GA and applications (Cotta, Mendes et al. 2003). Nowadays, researchers use the term evolutionary computing or evolutionary algorithms to cover the developments of the last 20 years.

Currently, the field of evolutionary algorithms has these main strands:

- Genetic Programming
- Evolution Strategies
- Genetic Algorithms
- Gene Expression Programming
- Differential Evolution
- Hybrid methods

These techniques were inspired by the main idea of genetic algorithms and have been developed separately and merged in the early nineties under the name of evolutionary computation and, more specifically evolutionary algorithms.

Not all of these methods will be discussed, as some of them are not directly related to this thesis topic. Simply, the basic ideas and methods in the field of genetic algorithm, genetic programming and gene expression programming will be reviewed, and then memetic algorithms will be discussed broadly.

4.2 Genetic Algorithms

Genetic algorithms were created based on the theory of evolution and genetics in biology. From another point of view, a genetic algorithm is a type of global search technique, which is used in computing to find true or approximate solutions. As a meta-heuristic algorithm, genetic algorithms in comparison with several other techniques such as greedy search can often find the best or nearly the best solution in a global answer space and usually do not get stuck in a local solution. GA is likely to find near the best solution for a given problem; however, GA process can be time consuming especially when it is applied to a large dataset. Therefore, nowadays in some applications researchers apply parallelism to speed up the processes.

The basic idea behind genetic algorithm can be described as follows. The problem is modelled by a representation of individuals. An individual is a possible solution in the space of solutions for the problem. In most implementations individuals are represented by bit strings. To evaluate the fitness of an individual to a problem, a fitness function is defined according to the characteristics and the goals associated with the problem. To find an optimum solution to a given problem, the following operations can be performed on the individuals to make them move towards better solutions:

- **Initialization:** during this process some individual solutions called chromosomes that bear properties known as genes are randomly generated.
- **Selection:** after producing an initial population some of the best chromosomes (solutions) are selected and an initial generation is formed.
- **Reproduction:** in this process, the next generation is produced from the initial one. To do this, two operators are applied which are borrowed from the nature: crossover and mutation.
    - **Crossover:** is a process in which some chromosomes are chosen (based on a probability) and broken down into two or more parts and are then merged together to produce new chromosomes.

- **Mutation:** is a process of flipping over one or more bits of chromosomes, again based on a probability.

After such operations, some of the best individuals are again selected from the current population and the process is repeated.

- **Termination:** the second and third steps are iteratively repeated until a termination condition is reached. The common terminating conditions are i) finding satisfactory minimum criteria, or ii) achieving a fixed fitness value, or iii) a fixed period of time has elapsed.

A Pseudocode of a typical GA algorithm is presented in Figure 4-1.

```
Begin
INITIALIZE PopulationOfSolutions;
EVALUATE each individual;
    Repeat Until (TERMINATION CONDITION is satisfied) Do
    SELECT parents;
    RECOMBINE to produce offspring;
    MUTATE offspring;
    EVALUATE offspring;
    SELECT individuals for next generation;
endDo
End.
```

Figure 4-1 Genetic Algorithm Pseudocode

In practice, there are a large number of variations in the implementation of a GA and what works in one case may not work in another. Therefore, some researchers looked for a way to predict algorithm performance for particular classes of problems. Reeves (1993; 2003) suggested that a population size of M with chromosomes size l is needed to get the probability of $P= (1 - \left(\frac{1}{2}\right)^{M-1})^{l}$ which show us how confident we are in exploring the

whole search space when we have binary representation. For example, a population size 17 is enough to get a probability of 0.99 for a string length 50.

4.3 Introduction to Genetic Programming

Genetic Programming (GP) was suggested by Koza (1992). The intuitive idea of GP is to generate computer programs automatically. In a GA we find optimal variable values so that their objective function has the maximum value. This kind of optimization problem is called parameter optimization. Sometimes problems are more complicated and we need to find the structure. Then we do structural optimization. This is when we use GP to find the function itself not only its parameters.

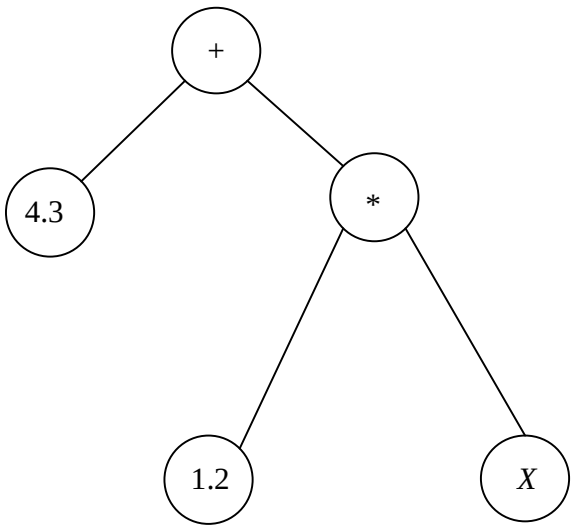


Figure 4-2 An example of a chromosome in Genetic Programming

Genetic programming is a kind of evolutionary algorithm where each individual is a computer program. These programs are usually in the form of trees which are the expressions of a predefined formal language. GP evolves programs which are in the form of algebraic expression to find the best expression tree (function). The predefined formal language represents a function.

In general, GP chromosomes have a variable length. In such a condition is relatively hard to determine crossover and mutation operators based on a prefix expression. Simple

crossover and mutation might generate illegal offspring. Therefore, fixed length chromosome has been proposed to be able to use simple variation operators.

Genetic programming is a popular tool for regression and function approximation tasks (Riolo, Soule et al. 2008). Genetic programming may be used to create a functional description of data by discovering classification functions or by modelling dynamic processes as described by (Banzhaf 1998). Structure optimization or learning is a research field that has attracted a lot of attention. GP for differential equation solving is a fascinating research field (Yu and Gen 2010).

4.4 Introduction to Gene Expression Programming

Gene Expression Programming (GEP) is a new form of genetic programming and was first introduced by Ferreira (2001). GEP is similar to genetic programming in that it evolves computer programs but the genotype and the phenotype are different entities (both structurally and functionally) and because of this, performance is improved. It has been shown in experiments to converge faster than older genetic algorithms (Ferreira 2002; Ferreira 2008). It also brings a greater transparency as the genetic operators work at the chromosome level (Wilson 2008). The most important application of GEP is in function finding and regression problems.

In the previous sections, we described genetic algorithm and genetic programming mechanisms. Here based on the previous understanding of the nature of a genetic algorithm we will describe how the GEP mechanism works.

The fundamental difference between the genetic algorithm, genetic programming and gene expression programming resides in the nature of the individuals: in genetic algorithm the individuals are linear strings of fixed length (chromosomes); in genetic programming the individuals are nonlinear entities of different sizes and shapes (parse trees); and in GEP the individuals are encoded as linear strings of fixed length (the genome or chromosomes) which are afterwards expressed as nonlinear entities of different sizes and shapes (i.e. simple diagram representations or expression trees). We

will see in the following chapter how these differences make GEP more effective than GA and GP.

GEP uses fixed length linear strings of chromosomes as the genotype, and the phenotype is in the form of expression trees which represent a computer program (Marghny and El-Semman 2005). The trees are then used to determine an organism's fitness. The decoding of GEP genes to expression trees implies a kind of code and a set of rules which are simple. The set of genetic operators applied to GEP chromosomes always produces valid expression trees (ET).

The genes of gene expression programming are composed of a head and a tail. The head contains symbols that represent both functions and terminals, whereas the tail contains only terminals. For each problem, the length of the head *h* is chosen, whereas the length of the tail *t* is a function of *h* and *n* is the number of arguments in the function. *t* is evaluated by the following equation.

$$t = H(n-1) + 1 \tag{4.1}$$

Consider a gene for which the set of functions is F= $\{+,-,*,/,Sqrt)$ and the set of terminals is $T = \{a,b\}$. In this case n = 2; if we choose an h = 6, then t = 6 (2 - 1) + 1 =7, thus the length of the gene is 6 + 7 = 13. One such gene is shown below:

$$*.-.a./.*.sqrt.a.b.a.a.b.a.b$$

where "." is used to separate individual building elements, *sqrt* represents the square root function and a, b are variable names. The above is referred to as Kava notation, and the above string is called a K-expression (Li X, Zhou C et al. 2004).

4.5 Hybrid Methods

Based on the *No-Free-Lunch* theorem (Wolpert 1997), no single algorithm can be found which works well for all types of problems. In this theory, Wolpert and Macready (1997) said that all optimization algorithms such as Genetic Algorithms, Simulated Annealing,

and Hill Climbing have the same average behaviour over all problems. Therefore, if one algorithm is more effective on average for a subclass of problems, it works worse in the other classes. This theory claims that the comparison of algorithms is useless unless we consider a subclass of problems and make a comparison based on a particular problem subclass.

An algorithm can work efficiently in some parts of a problem, but could be inefficient for others. Also, it was shown that as more domain knowledge is used to develop hybrid optimization algorithms, more efficient and specialized algorithms can be produced.

This led researchers to create a combination of different algorithms to solve a broad range of problems and use the advantages of each algorithm to make a more robust and more generalized problem solver. Also, many complex problems can be decomposed into a number of parts, for some of those sub parts exact methods (or very good heuristics) may already exist therefore in such a situation it does make sense to use a combination of the most appropriate methods for different sub problems (Carr, Hart et al. 2002). Moreover, many problems have a set of constraints and local search or other heuristics can be used for "repairing" infeasible solutions generated by standard global search operators. This is often simpler and more effective than attempting to find a specialized representation and or a set of combinations to be sure about the feasibility of all offspring (Carr, Hart et al. 2002).

Algorithms can be combined in different ways; a simple combination is a static combination, which means that the structure of combinations in terms of how and when those combined algorithms work together is predefined and fixed. In the more advanced cases a meta-knowledge selector decides when it has to switch between algorithms and also even decides which types of algorithm is better to use for a given problem. The most important types of combinations, which were found in the literature, are a combination of Global Search Algorithms with Local Searches, which use both algorithms' strengths to make an efficient search algorithm. We are not going to review the broad range of combined algorithms, as it is out of the scope of this thesis, instead we focus only on the range of combined algorithms which use GA in their combination.

Since the establishment of GA as a powerful algorithm in the computer science area, there have been significant amounts of work in its combination with other operations, such as search algorithms and machine learning techniques to improve performance in real-world applications. This section aims to provide a brief review of such hybridizations mainly based on notes by Sinha and Goldberg (2003).

It has been found through research in the Evolutionary Algorithm (EA) area that GAs are good at exploring the search area to find possible solution regions, but suffer from a lack of ability at finding a fine grained solution. As a result, it has slow convergence while performing that stage. On the other hand, local searches and problem-specific methods (such as heuristics) can act very well for this purpose. Previous work has shown that hybrid algorithms outperform a pure GA in most of the real world problems. Hybrids have been shown to be not only good in achieving a quality solution in a minimum time, but also in finding maximum solution quality in an acceptable time. Therefore, they are very successful from an optimization perspective.

In this thesis, we used hybrid evolutionary techniques in our first and second approaches; therefore, in the next section, we will provide an introduction to hybridization for a better understanding of the purpose and benefits of the hybridization in evolutionary computation.

The following categorization of Hybrid GAs will be reviewed from three perspectives (Hart, Krasnogor et al. 2005):

- Purpose of hybridization
- Hybrid architecture
- Types of secondary methods

4.5.1 Purpose of Hybridization

Exploitation

This class uses local search techniques for the purpose of fast convergence of algorithms after the GA has found a promising area in the search space. These include hybrids with Hill-Climbing and Simulated Annealing and Tabu search.

Repair

This class uses hybrids in the form of local searches for the purpose of repairing an infeasible solution. These hybrids are valuable in cases with highly constrained search spaces. In these cases, in a pure GA, a special kind of operator is used to avoid the generation of infeasible solutions, but this can be very hard to design.

Parameter Optimization

This class uses GA to optimize parameters in the second method, for example GA hybrids with neural networks for training of network parameters, or with reinforcement learning for choosing suitable parameters, or with Fuzzy logic for generating rule sets and membership functions.

GA Functionality Enhancement/Substitution

The second method can be used to perform some function of GA or enhance the performance of GA through better control of the process, for example, in the use of neural networks as a fitness estimator and the use of fuzzy logic controllers for dynamic control of GA parameters.

We will go through the categorization of local search combined with GA in detail further when memetic algorithms are discussed.

4.5.2 Architecture

We can classify hybrids based on how and when the secondary method is utilized. Combinations found so far can fall into one of the following categories:

- Pipelined Hybrids
- Asynchronous Hybrids
- Hierarchical Hybrids
- Embedded Hybrids

Pipelined Hybrids

In this group, there are two distinct stages for running the second method with GA. The possible combinations are in the form of Pre-processor, Postprocessor, and staged. A visualization of these three architectures is shown in

Figure 4-3.

In (a) GA is used to find a promising area in the search space, and then the second method continues until termination. In (b) the second method provides a good initialization for GA. Case-based reasoning and Tabu search have been used commonly for this purpose (Ramsey and Grefenstette 1993; Vilcota and Billaut 2008). In (c) there is iteration in performing GA and the second method. For example, each offspring which is selected based on having good fitness can go through a local search process for a specified duration, or to find a better solution in its neighbourhood. Then the process of evolution can continue again. This type of combination is the most common among those three. The subject of this thesis is based on this type of combination.

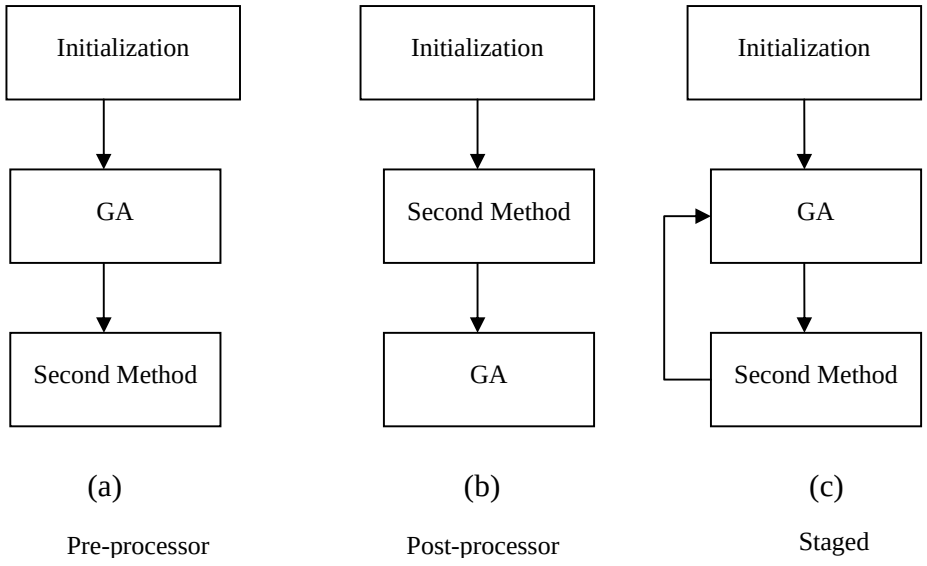


Figure 4-3 The three different possible combination for pipelined architecture (Sinha and Goldberg 2003)

Asynchronous Hybrids

This is a type of cooperation between two methods. The result of each may be utilized by another. The two methods typically have a shared memory to make solutions available for others. If the performance of the search is not good, they come back to the previous solution.

Hierarchical Hybrids

In this class, there is a procedure with multiple levels of optimization and at least one of those is a GA. An example of this class has been mentioned in work by Rogers (Rogers 1991). It uses a hierarchical approach for function approximation using splines with two levels of optimization. In the beginning, a GA is used to choose an appropriate basis function, and then linear regression is used for choosing coefficients with the least error for that function.

Embedded Hybrids

In this class, the secondary method is embedded inside of GA. The possible ways are:

*Initialization and seeding GA: F*or example, Case-Based Reasoning (CBR) has been used for preparing for GA a good initial population of feasible solutions.

Fitness evaluation: In some situations, the fitness function is not available and so a model of the system can be used for the evaluation of solutions. Using neural networks for this purpose is very popular.

Crossover: In some problems common crossover operators produce infeasible solutions. In these situations, problem specific operators are used to generate feasible offsprings from parents.

Mutation: This is a common example of an embedded solution in GA as mutation is the GA's Achilles heel. Local search in the form of G bit improvement of hill-climbing has applied for this purpose (Sinha and Goldberg 2003).

Special operators: A few studies have used a unified operator which integrated crossover and mutation. They can be categorized as special operators.

4.5.3 Secondary Methods

A wide range of existing machine techniques and algorithms have been used as a secondary method in combination with GA. Some of them are: Simulated Annealing, Local search methods, Artificial Neural Networks (Determining ANN architecture, Training of ANN, Selection and Generation of Training Data, ANN models for fitness evaluation, Input Feature Selection). Fuzzy logic, Tabu search, Decision Tree, Expert systems, Dynamic Programming, Case-based reasoning, Constrained Logic Programming, Branch and Bound.

4.6 Introduction to Memetic Algorithm

Over the past decades, Evolutionary Computation techniques have been successfully applied to various optimization problems in the field of engineering, biology, physics, chemistry, management, and computer science. However, it has been shown that they are not well suited to finetuned searching in complex spaces (Goldberg 1989). Moreover, other domain specific optimization techniques sometimes can outperform them, although these techniques do not have the same generality as Evolutionary Algorithms. These factors led to hybridization of Evolutionary Algorithms with other techniques to make them more efficient (Merz 2000).

There was some work on hybrid methods in the late 1980 that led to the memetic algorithms idea. The main pioneer work that led to MA was done in 1989 by Moscato and Norman (1989). In their paper, the authors introduced a technique which combined Genetic Algorithms with Simulated Annealing. In fact, Simulated Annealing played the role of a local search for GA. The initial motivation was to find a way to overcome the limitation of both techniques by combining them to solve a minimum Euclidean Travelling Salesman problem. The authors stated that the origin of the idea came from computer game tournaments used to study *the evolution of cooperation*. One year later, Moscato and Norman found out that several researchers previously used heuristics to improve solutions before recombining them. Particularly in GA, several works introduced problem domain knowledge in several ways. Finally, in 1989, Moscato introduced the term Memetic Algorithms (MAs) in 'On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Toward Memetic Algorithms' (Moscato 1989). He applied this algorithm to the Travelling Salesman as a representative test-bed. In this work, he suggested that using cultural evolution can be a better metaphor for avoiding constraints in biological evolution. In other words, he argued that some characteristics are not inherited from generation to generation; this kind of fashion or cultural transmission cannot be inherited by genes, for example, technological evolution or martial arts. Therefore, he claimed that a part of the natural process is missing and this

concept has not been implemented in Genetic Algorithms. He discussed that there is another flow of information which facilitates cultural transmission. The subunit in cultural transmission which transmits the cultural information is called memes.

The concept of a *meme* was first introduced by Richard Dawkins in his best-selling book 'The Selfish Gene' as the *unit of imitation* similar to a gene, but in the field of cultural evolution (Dawkins 1976).

There are differences between a gene and meme. First of all genes can be biologically inherited, but memes cannot. The second, and most important, difference between genes and memes is that before a meme is passed on, it is typically adapted by the person who transmits it by thinking and understanding, whereas genes get passed on entirely unaltered (other than by mutation). The third difference is that memes pass through generations for their popularity and generality, whereas genes survive passing through generations based on their strength and fitness. The fourth difference is that a meme can evolve much faster than a gene and the process is less resource consuming. The fifth difference is that meme evolution variation usually is not the product of chance or random exchange and usually is a goal-oriented process. For example, in science the raw combination of ideas does not lead to improved theory, usually scientists recombine ideas with their own ideas; therefore, this process allows innovation. In contrast, gene evolution does not include any idea about innovation and works based on pure chance. Therefore, the meme evolution is a goal oriented process, while gene evolution is an open-ended process.

4.6.1 Formal Definition of Memetic Algorithm

Ten years later after Moscato and Norman's paper, MAs became a well-known and useful approach for solving several NP-Hard optimization problems.

Different definitions have been introduced so far for memetic algorithms in the literature. In this thesis, we look at two common definitions as follows:

Definition 1: The Memetic Algorithm was inspired by the model of adaptation in natural systems that combine the evolutionary adaptation of population (already existing under the name Evolutionary Algorithms) with individuals' learning during their life time (in the form of local search) to make a more robust and efficient optimization algorithm. Memetic algorithms are sometimes called hybrid GA or scatter search (Carr, Hart et al. 2002).

Definition 2: This definition was given by Krasnogor in (Carr, Hart et al. 2002) which says memetic algorithms were inspired by Dawkins Meme which represents a unit of cultural evolution that can evolve to refinement. So far, memetic algorithms have not been used in this sense. The general definition of a memetic algorithm is presented in the Pseudocode in Figure 4-4.

```
Begin
Initialize Population Of Solutions;
Improve Individuals via local search;
Evaluate each individual;
Repeat Until (Termination Condition is satisfied) Do
Select parents;
Recombine to produce offspring;
Mutate offspring;
Evaluate offspring;
        Improve offspring via Local Search;
Evaluate offspring;
        Select individuals for next generation;
End Do
End.
```

Figure 4-4 Pseudocode for a memetic algorithm

There is much variation in this schema about when and how to apply local searches. Sometimes, a local search can act instead of a mutation operator. But, using the knowledge or problem domain is not an optional process; in fact it is a principal feature of MA. The search strategy behind MA differs from other evolutionary algorithms and,

as can be seen, MA incorporates knowledge of the domain in the form of a local search and, as a result, has some advantages over evolutionary algorithms. Although, most work on MA is about combinations of evolutionary algorithms with domain knowledge in the form of local searches, MA definition and scope are much broader than this and are not limited to only evolutionary algorithms.

4.6.2 Why MA is successful?

MA encompasses a broad class of metaheuristics. The success of MA is due to the synergy of different search approaches. Yet, why does the combination of different algorithms work so efficiently?

This is based on a strong theoretical background which originates from the *No-Free-Lunch* theorem. In this theory, Wolpert and Macready (1997) said that all optimization algorithms have the same average behaviour over all problems. Therefore, if one algorithm is more effective on average for a subclass of problems, it works worse in the other classes. An algorithm can work efficiently in some parts of a problem, but could be inefficient for others. Also, it was shown that as more domain knowledge is used to develop hybrid optimization algorithms, more efficient and specialized algorithms can be produced. Therefore, the main reason for MA success is the nature of MA which combines different algorithms. Another reason is the fact that MA uses domain knowledge in the form of local search.

Many complex problems can be divided into some parts for solving which an exact method already exists. In these cases, using the combination of appropriate methods is desirable. For example, Evolutionary Algorithms can be used as a part of the process and other algorithms to run as a past or pre-process which is the idea behind MA.

GA Compared to MA

In particular, to compare MA against GA we can point out following differences: In theory, a GA is useful particularly in global search cases in which there is not so much

knowledge from the domain, but in practice we use GA in cases which we have some knowledge from the domain. In other words, by applying just GA we do not use this knowledge to make our solver efficient (Hart, Krasnogor et al. 2005) and we know that the performance of an algorithm really depends on the amount and quality of problem knowledge. MA uses domain knowledge in the form of local search; therefore, can perform more efficiently.

Another reason for the success of MA relies on drawbacks of the GA approach. Although GA is very good in identifying suitable areas of the search space (exploration), they are less good at refining the near-optimal solution (exploitation)(Hart, Krasnogor et al. 2005). For example in *Recent Advances in Memetics*, Krasnogor mentioned that when GA is applied to the 'One-Max' problem, near-optimal solutions are quickly found, but convergence to the optimal solution is slow due to the fact that the choice of gene's mutation is random. Therefore, GA hybrids such as MA provide a more effective search by incorporating a more systematic search in the neighbourhood of good solutions.

The third reason for the success of MA compared to GA is that many problems have a set of constraints associated with them. In these cases we need special representation and operators to make sure that we produce feasible solutions. Local searches or other heuristics can be used for repairing infeasible solutions generated by other global search methods. This is more effective than finding a special type of representation or variation of operators to make feasible offsprings (Hart, Krasnogor et al. 2005).

As a result of these characteristics, Memetic algorithms are more efficient than simple GA, as they can reach an optimal point in a fewer number of generations, although a study shows that even having a fewer number of generations does not mean it works faster and could be in some cases more time consuming than GA (Areibi, Moussa et al. 2001).

Memetic algorithms are better at finding the optimal solution as they apply a local search to the final solutions found by the evolutionary process to fine tune it, while GA does not

guarantee finding the optimal solution. Therefore, the quality of solutions found by MA is usually better than GA. (Due to the drawback of GA in exploitation).

Simple GAs do not scale well with an increasing problem size, while MA can scale up better especially in some combinatorial problems in which solutions are nearby such as Travelling Salesman (Merz 2000).

MA has a disadvantage in comparison with GA, which is that it is prone to converging early and we need to think about a strategy to prevent early convergence.

4.6.3 Local Search

A typical local search can be described with the Pseudocode presented in Figure 4-5.

```
Begin
    /* given a starting solution i and a neighborhood function n */
    set best = i;
    set iterations = 0;
    Repeat Until (depth condition is satisfied) Do
            set count = 0;
            Repeat Until (pivot rule is satisfied) Do
                    generate the next neighbor j E n (i);
                    set count = count + 1;
                    If (f (j) is better than f(best)) Then
                            set best = j;
                    endIf
             endDo
             set i = best;
             set iterations = iterations + 1;
    endDo
End.
```

Figure 4-5 Pseudocode of a local search algorithm (Hart, Krasnogor et al. 2005)

Three components are important in local search algorithms.

Pivot Rule: This defines the condition for accepting an improved point. A greedy ascent pivot rule ends the inner loop as soon as an improvement is found, while steepest ascent pivot rule ends the inner loop when the entire neighbourhood has been searched. Sometimes when the neighbourhood is too large, it is recommended to use a random sample size which is considerably smaller than the original neighbourhood.

Depth of Local Search: This defines the termination condition for the outer loop. This parameter determines how many improving steps (iteration) have to be applied. There are considerable studies on the effect of changing this parameter as mentioned by Hart and Krasnogor (Hart, Krasnogor et al. 2005).

Neighbourhood Generating Function: The choice of an appropriate neighbourhood structure is important as it seriously affects the performance of the local search algorithm. It usually has to be done in a problem specific way by defining the set of solutions can be reached from s in one single step of a local search algorithm. Typically, a neighbourhood structure is not defined by explicitly pointing to a set of possible neighbourhood points, but rather implicitly by defining the possible local changes that may be applied to a solution.

The neighbourhood structure can also be represented as a graph, called the neighbourhood graph, G (v, e); V is the set of vertices or nodes of the graph that represent the solutions or points in the search space and two solutions are connected by an edge if they are neighbours (a set of edges show the move operator). Thus, a typical local search algorithm can be represented as a walk on the neighbourhood graph. This function defines a set of points after applying some move operators to the current point. Therefore we can consider the graphs defined by different move operators as fitness landscapes. Merz and others presented a number of statistical measures for characterizing fitness landscapes (Merz and Freisleben 1999; Merz 2000).

Local search algorithms may start from a randomly generated solution. In general, the solution found by a local search algorithm will not be a globally optimal solution; it may only be guaranteed to be optimal with respect to local changes. Certainly, the solution

quality obtained by a local search algorithm increases with larger neighbourhood size. A neighbourhood which guarantees that every local optimal solution is also a global optimum is called exact (Aarts and Lenstra 2003). Unfortunately, such neighbourhoods are typically of exponential size and searching for an improved neighbouring solution may take exponential time in the worst case. For practical reasons, it is required that each step of local search can be done in polynomial time (Stuetzle 1998).

4.6.4 Local Search in MAs

Here we are going to study local search inside of the MA context and see the effect of changing its parameters on MA performance.

There is a lot of variety about when and how to apply local searches and this is the most important factor in the design of a MA. As it was mentioned in section 4.4 (Hybrid Methods), the hierarchy of applying local searches inside of a MA is an important design factor, but the choice of local search algorithms and its parameters inside the local search is important as well.

Merz studied five famous problems in combinatorial optimization NK-landscapes, the Travelling Salesman, Binary Quadratic Programming, Graph Bipartitioning Problem and Quadratic Assignment Problem. He showed that MA on average outperforms other heuristic methods. He tried different types of local searches over those five problems and measured their performance. He also proposed some new local search algorithms in each problem which worked more effectively than other known local searches in those specified problems (Merz 2000). In TSP, there is a variety in types of instances, but in general it shows that lower correlation between tour lengths and distance to global optimum make a case hard and, also, a higher correlation between tour lengths and distance to global optimum indicates that recombination based search algorithms work well in these cases. MA with the newly proposed greedy recombination operator has been shown to outperform all its competitors: MAs with DPX or MPX recombination, MAs with non-sequential fur change mutation, and iterated local search. MAs with DPX (the distance preserving crossover operator) and GX (A greedy recombination operator)

recombination and mutation have been applied to various instances contained in TSPLIB[4] to show robustness and scalability of the approach. The memetic algorithm appears to be superior in average solution quality and running times. Finally, the MA with GX has been applied to very large instances of up to 85900 cities and is, thus, the first meta-heuristic known which can tackle very large problems. In Graph Bi-partitioning he proved that for geometric graphs, the combination of differential greedy and Kernighan-Lin local search is sufficient for small graphs (up to 5000 nodes).

There have been a large number of studies over the different statistical measures of landscape for prediction problem difficulty. Merz and Freisleben (1999) studied some of these measures and showed that choice of move operators can have a strong effect on MA. In another study, a number of statistical measures for characterizing fitness landscapes was presented (Hart, Krasnogor et al. 2005). However, there is still a huge amount of potential for further study of landscapes analysis and performance measurement.

Merz mentioned that the choice of move operator (neighbourhood structure) has a dramatic effect on the efficiency of local searches and consequently on MA (Merz 2000). Krasnogor mentioned that in some cases domain knowledge can be used to guide the choice of neighbourhood structure(Hart, Krasnogor et al. 2005). Recently, it has been discovered that optimal choice of operator is not only specific within a class of problem, but also inside MA can be dependent on the state of the evolutionary search. Krasnogor (Hart, Krasnogor et al. 2005) also mentioned that changing the neighbourhood operator during a search, where points are locally optimal for a given neighbourhood operator, may result in a means of progression. Due to this fact, points that are locally optimal in one neighbourhood structure may not be in another structure (except those that are globally optimal). This fact leads to variable neighbourhood search algorithms (Hart,

[4] TSPLIB is a library of sample instances for the TSP (and related problems) from various sources and of various types

Krasnogor et al. 2005). In the following section, we present some of the design issues in MA, which involves how and when a local search has to be applied inside a MA.

4.6.5 Design Issues

The process of designing effective and efficient MA needs problem specific details and therefore, is an ad-hoc process. Krasnogor and Smith (2005) reviewed some examples of MAs application to provide a systematic model for MA which determines role of different parts and their relationship. They tried to provide a better understanding of how to design MA for a problem and also provided a conceptual framework to deal with difficult questions about MAs general behaviour.

They raised several important questions which must be addressed to design a MA effectively.

- The first and foremost of those is the question of what is the best trade-off between local search and global search?
- Where and when local search can be applied within the evolutionary process?
- Which individuals in the population should be chosen for improving by local search?
- For how long should perform local search?
- How genetic operators should be integrated with local search to get an effective combination?

They indicated that these theoretical questions have not been answered properly yet in the MA literature.

4.7 Summary

In this chapter, we provided an introduction to evolutionary computation including genetic algorithms, genetic programming and gene expression programming. We also explained combined evolutionary methods and different methods for combination of

algorithms. Then we provided a good background on memetic algorithms which is a combined genetic algorithm. We discussed the superiority of memetic algorithm over the standard genetic algorithms.

The technique that we used for our first approach is a hybrid gene expression programming. Our second approach uses a hybrid genetic algorithm. Therefore, we considered to provide a solid background for the reader in evolutionary computation and hybridization in order to understand those approaches.

Now that the reader has been provided with enough background, in the next chapter, we will review our first approach which uses a hybrid of Gene Expression Programming with local search to solve a system of differential equations used for modelling a GRN.

Chapter 5

Approach 1: Memetic Gene Expression Programming

"What distinguishes a mathematical model from, say a poem, a song, a portrait or any kind of 'model', is that the mathematical model is an image or picture of reality painted with logical symbols instead of with words, sounds, or watercolors."

-John Casti, Reality Rules

5.1 Introduction

In the previous chapter we provided background information on evolutionary computation techniques such as Genetic Algorithms and Gene Expression Programming and combined evolutionary algorithms called memetic algorithms. That information was necessary in order to understand our first and second approaches which employed evolutionary techniques. In this chapter, we introduce our first approach to solve the GRN inference problem which will propose a new combined evolutionary technique called Memetic Gene Expression Programming (MGEP). The MGEP is a combination of Gene Expression Programming (GEP) with a local search method. GEP is known to be a highly effective tool for function approximation and regression; however when it comes to parameter estimation it is not as effective. Previously, local search methods combined with other evolutionary techniques such as genetic programming proved to be an effective tool for parameter estimation. Therefore we used a local search mechanism combined with GEP to make a more effective tool for function approximation. We applied our Memetic Gene Expression Programming technique to an artificial dataset to obtain a system of differential equations which were used for modelling a gene

regulatory network. We also investigated the effect of noise on our algorithm, as this is an important consideration for any algorithms applied on microarray data which is high in noise and the number of missing values.

In this work, for the first time, a combination of Gene Expression Programming with a local search method is proposed and proved to be effective compared to GEP alone. This combined method also showed robust behaviour in the presence of noise and missing values.

We provided an introduction to Gene Expression Programming (GEP) in Chapter 4 Section 4.4. Here, we will firstly describe how this technique can be used for GRN discovery and then we will describe our proposed approach which is based on GEP.

5.2 Gene Expression Programming for GRN Inference

Gene Expression Programming is a highly effective method for function finding. This quality of GEP makes it a suitable tool for any application which requires function approximation. One common method for GRN inference involves approximation of differential equations.

Temporal differential equations are the most common modelling used to build a Gene Regulatory Network (GRN) from time series data (Wang, Joshi et al. 2006; Hallinan 2008). Differential equations are a powerful and flexible model to describe complex relations among components. Differential equations represent a GRN by quantifying the rate of changes of the expression of one gene as a function of the expression of the other genes.

A system of differential equations which can be used to model a GRN is shown in Equation 5.1:

$$\frac{dX_i}{dt} = f_i(X_1, X_2, \dots, X_n) \ \ (i = 1,2, \dots n) \tag{5.1}$$

Here X_i is the expression level of the *i*-th gene (state variable) and *n* is the number of the genes (components) in the network.

Evolutionary computation is a particularly useful approach when a problem cannot easily be solved mathematically and we cannot realistically look for an optimal solution, but one or more good solutions are needed. Therefore it is particularly suitable for the problem of solving a differential equations system. Different kinds of evolutionary computation techniques have been applied to this problem, ranging from extensions of genetic algorithms to genetic programming and differential evolution.

Sakamoto and Iba (2001) used genetic programming (Koza 1992) to solve this problem modelled by a system of differential equations. Solving the general form of a system of differential equations is very difficult, so a fixed form, called the *S-system* (Savageau 1988), was used and the goal became simply to optimize the parameters in the fixed equations. An S-system is a type of power-law formalism. The concrete form of the S-system is shown in Equation 5.2 where X_i is a state variable. The first term gives us all the effect of increasing $X_i$whereas the second term gives the effect of decreasingX_i.

$$\frac{dX_i}{dt} = \alpha \prod_{j=1}^{n} X_j^{g_{ij}} - \beta \prod_{j=1}^{n} X_j^{h_{ij}} \quad (i = 1,2, \dots n) \tag{5.2}$$

The first work which used genetic algorithms to solve the S-system was presented by Maki et al. in (2001). There are other works which applied genetic algorithms to this problem such as a study by Morishita (2003) which used a genetic algorithm to find parameters for an S-system representing a 5-node network. Kikuchi and others (2003) at the same time reported a good result for the same number of nodes. Spieth and others (2004) used a genetic algorithm along with evolutionary strategies as a local search method. Later on, in 2005, genetic programming was used to solve the S-system by Matsumura et.al. (2005) and appropriate solutions were obtained. Also in 2005, for the first time, differential evolution was used for this purpose by Noman and Iba (2005). Their work exhibited high performance. However in their study the number of genes was still limited to five and the model could not easily be scaled up for larger networks. The

reason for this is the fact that the number of parameters in a differential equation system is proportional to the square of the number of genes in the network. Therefore, when the number of genes increases, the algorithms must simultaneously estimate a larger number of parameters. This is why inference algorithms based on the differential equations model have so far only been applied to small-scale networks of about five genes.

At the time of writing this thesis we found a new publication which suggests using global search methods (Genetic Programming) for finding the general structure and then a local search method (RLS method) for finding parameters (Wang, Qian et al. 2010). They also used a simplified S-system for modelling the network in this way they reduced the number of detailed parameters. As a result, they successfully scaled up the size of the network for which they could estimate the simplified S-system it might be worthwhile to try their simplified S-system and RLS method within our proposed approach which uses a more powerful global search method (Gene Expression Programming).

Evolutionary techniques were used along with other modelling approaches for gene regulatory networks. An example of that is a study by Eriksson and Olsson (2004) which used genetic programming to successfully solve a Boolean Network of 20 genes.

Here, we attempted at solving the problem of inferring a gene regulatory network modelled by a system of differential equations with an extension of the Gene Expression Programming (GEP) algorithm. GEP has been applied in many regression problems successfully. In particular, it was used previously for solving elliptic differential equations (Jiang, Wu et al. 2007). Our extension exploited the effectiveness of GEP in finding the structure of gene regulatory network modelled by ordinary differential equations. It also used a local search technique along with GEP for extra benefits. The combination of these methods, using GEP as a global search for finding a function structure and a local search for fine tuning the model parameters, resulted in a more powerful algorithm.

The combination of global search methods with problem specific solvers is known as Memetic Algorithms (MAs) (Moscato and Norman 1989). This combination has been proved to be effective as the global search tries to find the best solution for the problem,

and meanwhile the local search tries to improve the solution by finding the best solution in a neighbourhood. Usually global search techniques find near to optimal solutions, not the optimal one. Using local search helps to improve the solutions found by global search. The problem-specific solvers are usually implemented as local search heuristic techniques. The hybridization is meant to accelerate the discovery of an optimal solution or to reach a solution which is impossible to discover by either of the component methods (Krasnogor, Aragón et al. 2006). So far, conventional genetic algorithms have mainly been used in MAs as the global search method, however, the scope of MAs is not limited to the genetic algorithms and in general any global search method can be used (Krasnogor & Smith, 2005). In Spieth et al. (2004) authors used a genetic algorithm along with evolutionary strategies as a local search method to solve an S-system which was used for modelling of a gene regulatory network. Sakamoto and Iba (2001) used a local search algorithm along with genetic programming to obtain the constant parameters of the target function effectively. Here for the first time we have proposed an MA with GEP as the global search method. The Least Mean Square method (LMS) was used as the local search method. We have used the same data as were used in a previous study in the literature (Noman and Iba 2005) and compared the efficiency of our method with conventional genetic programming.

5.3 Memetic Gene Expression Programming for GRN Inference

Here we present an algorithm designed to infer a gene regulatory network from the observed time series data. As noted earlier, the problem can be modelled as a set of differential equations. We used a GEP algorithm to evolve the structure of the gene regulatory network and to find the best form of differential equations from the observed time series of the gene expression. Although GEP is effective in finding a suitable structure, it is not so effective in optimizing the parameters of the formula such as constants or coefficients. Thus we enhanced it by incorporating a local search process into GEP to find the constant parameters of the equations more effectively and we called

it *Memetic Gene Expression Programming*. Local search methods are known to be able to find the constant values and parameters effectively, and GEP is known to be effective in finding function structures. This combination results in an effective algorithm which is highly capable for function estimation. The overall algorithm is presented below:

1. The GEP evolution begins with the random generation of linear fixed-length chromosomes for individuals of the initial population.
2. In the second step, the chromosomes are translated into expression trees and subsequently into mathematical expressions, and the fitness of each individual chromosome is evaluated based on the formula presented in Equation (5.3) by using the Runge-Kutta method.
3. Local search is applied on individuals at some interval generations.
4. The worst individuals in the population are replaced with the improved individuals generated earlier.
5. These operations are applied: selection with tournament selection and then genetic recombination.
6. The above steps are repeated until there is no further improvement in the fitness function.

The local search algorithm was applied in two different ways. In the first method, it was used only for the best individuals in each generation. In the second method it was used on the whole generation at some intervals. The result of the second method was better than the first method; therefore, the reported results are based on the second method of applying the local search procedure.

5.3.1 Fitness Function

In general, the genetic network inference problem is formulated as a function optimization problem to minimize the following sum of the squared relative error and the penalty for the degree of the equations:

$$f_i = \sum_{i=1}^{n} \sum_{k=0}^{r-1} (\acute{X}_\iota\,(t_0 + k\,\Delta t) - X_i(\,t_0 + k\Delta t))^2 + \sum_{j=0} a_j b_j \tag{5.3}$$

X_i: the expression value of gene i
t_0: the starting time
∇t: the step size
n: the number of components in the network
T: the number of the data points

Where $X_i(\,t_0 + k\Delta t)$ is the given target time series (k=0, 1,…, T-1) and $\acute{X}_\iota(t_0 + k\,\Delta t)$ is the time series acquired by calculating the system of differential equations represented by a GEP chromosome. All of these time series are calculated using the Runge-Kutta method. This fitness function has often been used in previous studies in GP, for example by Samakato and Iba (2001).

The problem of inferring gene networks based on the differential equations has several local optima. Local optimal points are the points in the search space that are optimal within their local neighbourhood, in contrast with global optimum which is the general optimal solution. The high number of local optima happens because the degree of freedom of the model is high. The degree of freedom is the indicator of the number of variables required to estimate a model. As we are not interested in finding the local optimum instead of the global optimum, a penalty function has been introduced by Kimura et al (2004) to avoid the local optima solutions. This penalty function which is the second part of the fitness function encourages low degree solutions. The a_j is the penalty coefficient for the *j*-th degree and b_j is the sum of the absolute values of coefficients of *j*-th degree.

5.3.2 Local Search for the Local Optimizations of the Model

GEP is capable of finding a desirable structure effectively, but it is not very efficient in the optimization of the constant parameters, as it works on the basis of the combination

of randomly generated constants. Thus we used the Least Mean Square (LMS) method to explore the search space in a more efficient way. To be more specific, some individuals were created by the LMS at some intervals of generations. Therefore we used the LMS method to find the coefficient of the expression of the right-hand sides of the system of differential equations. We applied LMS in such a way that was previously introduced by Sakamoto and Iba (2001) in combination with GP.

Consider the expression approximation in the following form:

$$\mathrm{y}(x_1, \dots, x_l) = \sum_{k=1}^{M} \mathrm{a_k F_k}(\mathrm{x_1(i)}, \dots, \mathrm{x_l(i)}) \tag{5.4}$$

Where $F_k(x_1(i), \dots, x_l(i))$ is the basis function, $x_1, \dots, x_L$ are the independent variables, $y\ (x_1, \dots, x_L)$ is the dependent variable, and *M* is the number of the basic functions.
Let *a* be the coefficient vector and χ^2 as follows:

$$\chi^2 = \sum_{i=1}^{N} (y(i) - \sum\ a_k F_k\ (x_1(i), \dots, x_l(i))^2 \tag{5.5}$$

The purpose of the local search is to minimize the function in Equation (5.5) to acquire *a*. *N* is the number of data points. Let *b* be the vector *y(1),…y(N)* and *A* be a *N*N* matrix described as follows:

$$\begin{bmatrix} \mathrm{F_1}\left(\mathrm{x_1}(2), \dots, \mathrm{x_L}(2)\right) & \dots & \mathrm{F_M}\left(\mathrm{x_1}(1), \dots, \mathrm{x_L}(1)\right) \\ \mathrm{F_1}\left(\mathrm{x_1}(2), \dots, \mathrm{x_L}(2)\right) & \dots & \mathrm{F_M}\left(\mathrm{x_1}(2), \dots, \mathrm{x_L}(2)\right) \\ \dots & \dots & \dots \\ \mathrm{F_1}\left(\mathrm{x_1}(\mathrm{N}), \dots \mathrm{x_l}(\mathrm{N})\right) & \dots & \mathrm{F_M}\left(\mathrm{x_1}(\mathrm{N}), \dots \mathrm{x_l}(\mathrm{N})\right) \end{bmatrix} \tag{5.6}$$

y(i) for the *i*-th equation of the system is calculated as follows:

$$y(i) = \dot{X}_J\big|_{t=ti} = \frac{x_j(t_i + \Delta t) - x_j(t_i)}{\Delta t} \tag{5.7}$$

Then the following equation should be satisfied to minimize Equation (5.5).

$$(A^T A).a = A^T b \tag{5.8}$$

a can be acquired by solving Equation (5.8).

5.4 Experiments

To confirm the effectiveness of the proposed algorithm, we have used a small network model with four sets of time series data with different initial values. The number of the network components (genes) was considered to be five.

From those four experiments, here we present the results for one, which was the most complicated example. Figure 5-1 shows the gene network used in this experiment.

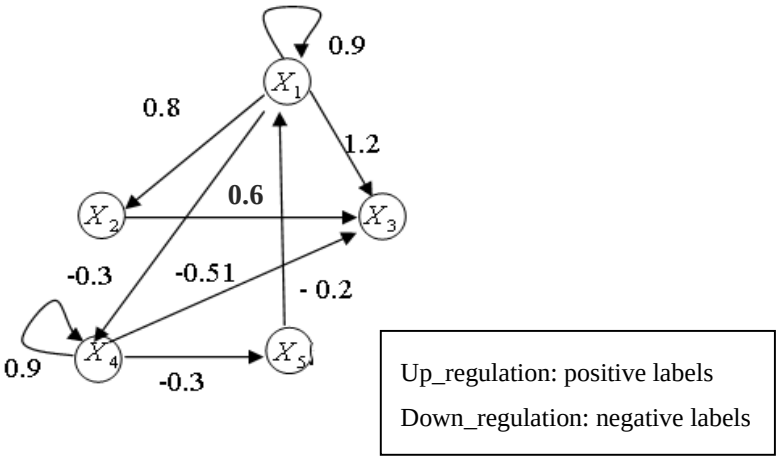


Figure 5-1 A sample of weighted Gene Regulatory Network

A weighted network was used to represent gene networks (Weaver, C.T.Workman et al. 1999). Each node is a gene and an arrow indicates a regulatory relation between two elements (gene). Negative values show an inhibition relationship and positive values show activation.

To account for the stochastic behaviour of GEP, each experiment was repeated for 20 independent runs, and the results were averaged. Table 5-1 lists the parameter values used for these runs.

Table 5-1 General settings of our algorithm

Number of generations	500
Population size	100
Mutation rate	0.044
One-point recombination rate	0.2
Two-points recombination rate	0.2
Gene recombination rate	0.1
IS transition rate	0.1
RIS transition rate	0.1
Gene transposition rate	0.1
Function set	+ - * /
Terminal set	α

Figure 5-2 shows the observed expression levels of the five components (genes) of the network and the predicted level produced by our method.

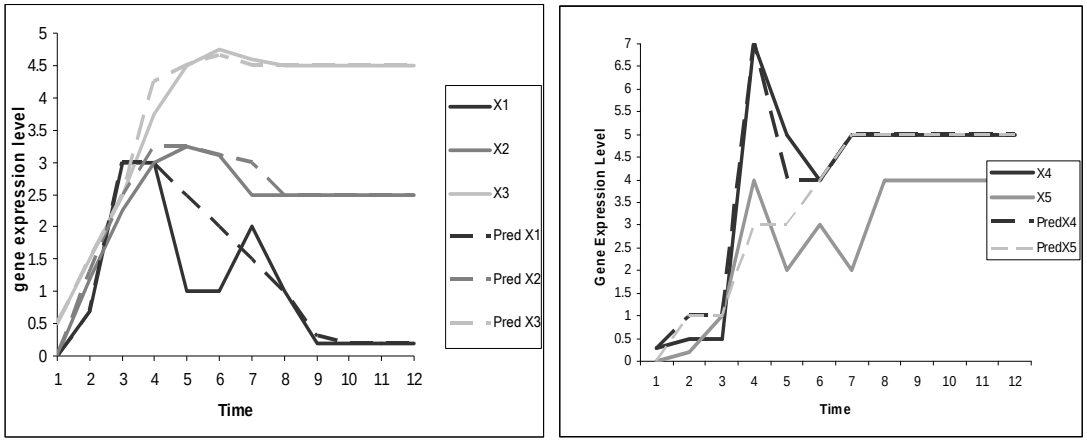


Figure 5-2 Predicted versus actual gene expression levels for the best model obtained

The effect of local search on the performance of the algorithm is presented in Figure 5-3. The local search was applied in two different ways: in the first way it was applied to the best individual of the generation and in the second way it was applied to the whole population. The first approach rarely improved the performance, but the second approach

significantly improved the fitness of average individuals in the population, especially in the early stages of evolution.

The reported result is based on the second approach to applying local search. It can be seen that on average the memetic system using both GEP and LMS achieved superior fitness levels than the system using GEP alone.

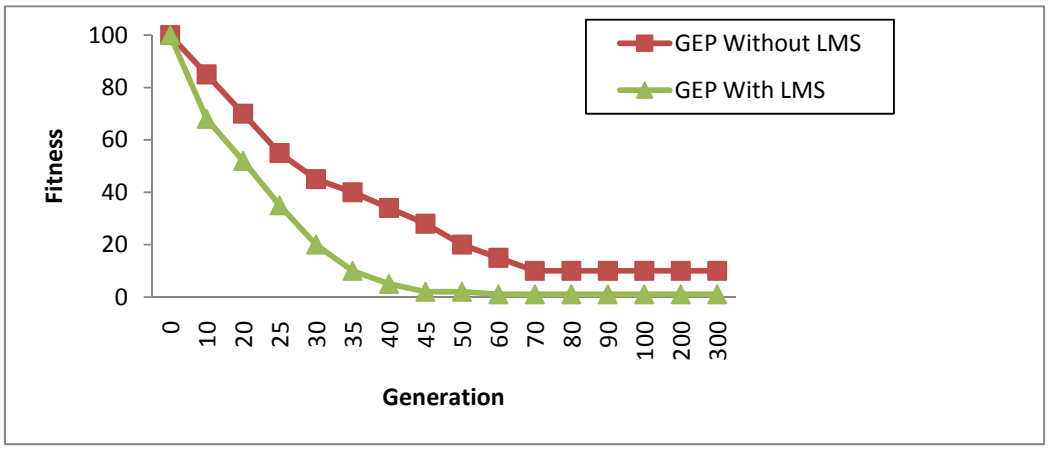


Figure 5-3 Comparison of GEP performance with and without local search

The local search (constant creation method) applied to the best individual of the generation can seldom improve them, however when it is applied to the whole population it can significantly improve the fitness of average individuals in the population, especially in the early stages of evolution.

We also compared our algorithm with the conventional GP algorithm. For this purpose we used GPLAB (MATLAB toolbox for genetic programming) with default parameter values. The result is presented in

Figure 5-4 which shows that the proposed method had a faster convergence rate by an index of 100 compared to the conventional GP.

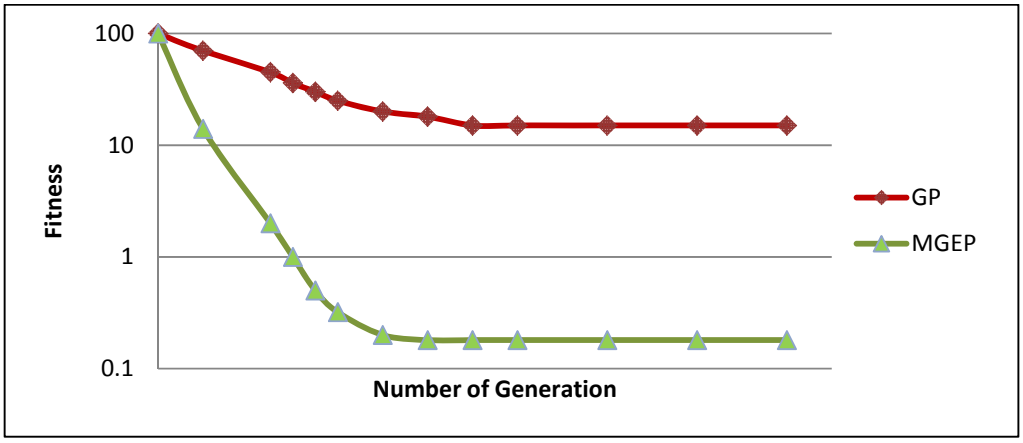


Figure 5-4 Performance comparison of MGEP against GP (logarithmic scale)

5.5 Effect of Noisy Data

We introduced artificial noise to the data to test the robustness of our method. Usually in microarray data the presence of missing values is a common problem causing many difficulties. Therefore we considered this type of problem here. We started with one missing variable per sample (2% noise) and then increased the amount of the missing variables up to 10% noise. The effect of such a problem is presented in Table 5-3. We present the correlation coefficient (r) that quantifies the similarity between predicted values and observed ones as the measure of robustness of the algorithm in the presence of missing values or noise.

In the second experiment, we tested the effect of Gaussian noise on the data by perturbing a certain value x_i with a random number drawn from a Gaussian distribution $N(0, \sigma_1)$ by $\dot{x_i} = x_i + \sigma_i N(0,1)$. We used a Gaussian noise because it means that the noise is normally distributed across experiments. This simulates a random noise. Table 5-2 shows the result of applying noise on the gene expression values.

Table 5-2 Effect of noise with adding missing values

Output	*R*
Output without noise	0.891
Output with 2% noise	0.846
Output with 10% noise	0.798
Output with 20% noise	0.702

Table 5-3 Effect of Gaussian noise

Output	*R*
Output without noise	0.891
Output with 2% noise	0.888
Output with 10% noise	0.863
Output with 20% noise	0.801

The results in Table 5-2 and Table 5-3 show that the noise in the form of missing values affects the algorithm more than the Gaussian noise.

The proposed system presents robust behaviour in the presence of noise, along with good performance. To compare the robustness of this algorithm in the presence of noise and also to make further investigation of the effect of noise type on our GEP system, we investigated the Gene Expression Programming (GEP) literature. It has been said that GEP is a robust method in the presence of noise, although there is not enough literature available on the effect of different types of noise on GEP systems. The only trace of this type of work is a study by Lopes and Weinert (2004). In this work, they used a simple form of random noise on each value and still obtained a good result. Therefore we decided to review the effect of the noise on Genetic Programming (GP) algorithms, as GEP can be considered as an extension of GP.

Typically, the fitness function for regression problems is based on a sum-of-errors, involving the values of the dependent variable directly calculated from the candidate expression. Although this approach is extremely successful in many circumstances, its performance can decline considerably in the presence of noise. Therefore, in a study by Imada and Ross (2008) it was suggested to use a feature-based fitness function in which the fitness scores are determined by comparing the statistical features of the sequence of values rather than actual values themselves. This type of fitness functions can be

considered for future research on improving the proposed algorithm in the presence of noise.

5.6 Analysis of the Results and Future Work

Recently, evolutionary computation methods have been used for model-based inference of gene regulatory networks. In this work, we have investigated the suitability of Gene Expression Programming (GEP) for this problem. We have also proposed a memetic version of GEP which uses LSM as the local search procedure to improve the quality of solutions. The experimental results reported in this chapter, using synthetic gene expression data; show that the proposed Memetic GEP (MGEP) algorithm can find a suitable combination of constants and function structures.

The proposed MGEP can be further examined with other local search methods to more effectively fine-tune parameters. It is also vital to increase the number of genes in the network to scale up this method as much as possible. In reality, the gene regulatory network usually has more than ten components. To the best of the author's knowledge, existing evolutionary techniques can not deal with this number of components considering real gene expression values. Partitioning is a possible solution to scale up these methods. Partitioning methods which have been previously used with other evolutionary algorithms by Kimura et al. (2004) have improved their scalability. Also as it was mentioned earlier using the newly proposed simplified version of the differential equations system (Wang, Qian et al. 2010) can help to overcome the scalability problem.

In order to study the effect of real noise on our algorithm, the noise in the real data needs to be mathematically modelled. In this way it would be possible to investigate the effect of real noise on our algorithm. The only part of the noise in our study which had a corresponding part in the nature was the missing values. Modelling of noise in the form of mutated values is subject to further investigation of the distribution of noise in real microarray data.

In general, we made an improvement on the state-of-the-art function approximation techniques in particular for the system of differential equations. However, the scalability

remained an open problem. The scalability problem is a common problem previously reported in the related literature of modelling GRNs with differential equations.

5.7 Summary

In this chapter, we explained our first proposed approach. This approach modelled a Gene Regulatory Network as a set of differential equations and tried to solve these equations using Memetic Gene Expression Programming. The Memetic Gene Expression Programming which was proposed for the first time in this thesis uses LMS (Least Mean Square) method as a local search mechanism. The local search mechanism improved the quality of parameter estimation considerably and helped the algorithm to converge faster and to a better solution. The algorithm was tested on an artificial dataset and was reported to perform considerably better than the Genetic Programming technique or simple Gene Expression Programming. The effect of noise is also studied in this work.

Despite achieving a higher performance compared with other techniques applied on differential equations, this approach still suffered from the lack of scalability in real-sized gene networks. The performance indicated that it will not be applicable for hundreds of genes and this is mainly due to using differential equations modelling which requires the approximation of many detailed parameters.

Chapter 6

Approach 2: Combined Evolutionary Algorithms

"The scientist is not a person who gives the right answers; he's one who asks the right questions".

~Claude Lévi-Strauss, Le Cru et le cuit, 1964

6.1 Introduction

The previous approach proved to be effective in improving the inference of Gene Regulatory Networks (GRN) modelled by a differential equations system; however, using that technique we cannot overcome scalability problems. Underlying the scalability problem is its modelling approach, which uses a system of differential equations. Using those equations we need to consider too many parameters and details which are computationally expensive. In reality, there is not much need for exact estimation of such parameters, as they are so variable, depending on the thermodynamic situations such as temperature and other environmental factors.

That is why we considered a coarse-grained equation-free modelling in the second approach, to be able to find a network of real size having hundreds of genes. There are several methods for such a modelling, including building a directed graph of dependencies. In such an approach we tried to build the network by finding edges and usually considering one edge at a time. Many studies have been done using this modelling such as Relevance network (Butte and Kohane 2000) and its extension CLR (Faith, Hayete et al. 2007), BioLayout Express (Theocharidis, Dongen et al. 2009) and ARACNE (Margolin, Nemenman et al. 2006).

Another important factor mentioned in the literature was using information from the domain knowledge whenever it is possible to improve the discovery process which helped us to limit the search space and discover real-sized networks. Using information from domain knowledge can be useful in any application, in particular for GRN discovery, as the data is noisy, the number of samples compared to the number of genes is so low and there is also a hidden temporal relationship between variables. The benefit of using domain knowledge has been demonstrated in the context of microarray analysis very well (Subramanian, Tamayo et al. 2005; Franke, Bakel et al. 2006).

A line of research related to using information from domain knowledge was initiated by a famous study called *Gene Set Enrichment Analysis* (Mootha, Lindgren et al. 2003; Subramanian, Tamayo et al. 2005). In this work a combined group from Harvard and Cambridge suggested a different approach to the analysis of microarray data. Their method takes a different perspective on the problem. The method starts with some gene sets known to be involved in a biological function, such as cell death (apoptosis), and determines whether they can identify any difference between a normal versus disease group. Those gene sets which significantly differ between the two groups are chosen as the most informative gene subsets. In this way, they involve existing knowledge about biological gene pathways to obtain a result which is biologically meaningful. The information about gene sets usually comes from Gene Ontology. Gene Ontology is a classification effort to organize information related to genes hierarchically in the form of gene sets, which are related to a specific function or a part of a cell (Ashburner, Ball et al. 2000).

The statistical method used in *Gene Set Enrichment Analysis* looks at the most coherent and modest changes in a group of genes instead of a dramatic change in individual genes. For example, a 20% increase in the expression level of a group of genes in the same pathway may be more important than a 20-fold increase in a single gene (Subramanian, Tamayo et al. 2005). Using this approach, their study detected a considerable number of genes in common between different studies related to diabetes that were previously not detected by earlier approaches.

This study has led to a new generation of tools for microarray analysis which uses biological knowledge from the beginning to find biologically meaningful changes in the data called functional gene set analysis (Al-Shahrour, Díaz-Uriarte et al. 2004; Al-Shahrour, Díaz-Uriarte et al. 2005; Lee, Braynen et al. 2005; Jiang and Gentleman 2007). Some of these studies tried to add more value by adding more information from the domain knowledge such as protein interaction networks to define a group of genes which are functionally related. Another group of studies has aimed to develop a statistical function to measure the changes more accurately for small-sized datasets when the number of samples is less than ten such as *roast* function (Wu, Vaillant et al. 2010) developed in the *limma* package (Smyth 2005).

Based on the above literature, we considered using similar ideas in the context of Gene Regulatory Network discovery. There are some previous studies which use the information about gene groups (modules) in the context of Gene Regulatory Network. There is a famous study called Module Network which uses transcriptional modules to partition the search space and uses Bayesian Network to find the structure of each module and the dependency of modules together (Segal, Shapira et al. 2003). In this way the algorithm was able to explore the search space which was not possible by the use of Bayesian Network on the whole search space, as Bayesian Network is not an applicable tool where a search space is too big. A limitation of Module Network is that it makes use of overlapping genes between modules to connect them. This implies we need overlaps for connections. The algorithm also has to identify one gene as the representative for each group to connect them. These assumptions decrease the similarity of the result with known networks.

In networks with a large number of nodes, a useful approach to reconstruction is to partition them and recompose the parts. Partitioning a network into sub-networks is meaningful and useful if the resulting sub-networks or modules are biologically relevant and display characteristics that are retained after decomposition and recomposition. Partitioning into modules has shown to be useful; however, it is still in its infancy (Kepes 2007).

Inspired by Gene Set Enrichment, we looked for a way to use gene set information to find the GRN more effectively. Based on the properties of the biological network and gene networks, which we will discuss in detail in Chapter 9, we know that the network is a modular network and is built from cliques. Those cliques are functional gene sets that cooperate to perform a function (Ravasz, Somera et al. 2002; Almaas, Vazquez et al. 2007). The fact is, these modules do not stay the same under different conditions such as disease conditions and may undergo substantial changes. Therefore, we can use them to find which modules have the most changes. Our idea of using modules is similar to GSEA but unlike GSEA it enable us to detect changes and find new modules whereas GSEA can only indicate the changes in the original gene sets and cannot point out the genes which are added or deleted.

Based on the above facts we considered a design that starts with the gene sets and evolves them to fit to the microarray data, then uses these final gene sets to build the overall network. In doing that we needed a method to evolve the gene subsets and explore the possible answers. A combinatorial search process can globally search the answer space. We can also use a local search method to efficiently improve the possible candidates (gene sets). A suitable tool for exploring the search space globally can be any combinatorial search. We chose genetic algorithm as it is easier to apply in any given problem without any consideration or assumption about the data and the nature of the problem.

Corne and Pridgeon (2004) have argued that genetic algorithms may be particularly well suited for reverse engineering of biological networks, which are themselves a product of an evolutionary process. Our familiarity with evolutionary techniques was an additional factor in choosing GA as a tool to implement such an idea. GA has previously been used in this context, especially to obtain a random Boolean network (Halinan 2008; Marbach, Mattiussi et al. 2009). We extended the conventional GA to be able to evolve a chromosome with a partial solution (gene sets) instead of a whole solution. Gene sets are basically the partial solutions that undergo the evolutionary process in order to be

refined. In the next stage these partial solutions were combined to build the final network (complete solution).

Similarity of the solutions with the domain knowledge is an important aspect of any algorithm in this area. Therefore, we found a way to make our solutions compatible with the domain knowledge. Our solution for this was to use a case retrieval mechanism as a local search process to extract information from domain knowledge in order to upgrade our partial solutions. The combination of GAs with the local search methods is known as Memetic Algorithm (MA). In the past, local search methods such as hill climbing and *Tabu search* have been applied to the solutions to upgrade them (Hart, Krasnogor et al. 2005). Here for the first time, we proposed using direct information from the domain knowledge as a case retrieval mechanism. The local search mechanism tried to make solutions similar to the domain knowledge as much as possible and find the shortcuts in the search space using the information known about the association of genes. An example of such information is protein interaction networks. Protein interaction networks give us a picture of how proteins which are in turn products of genes, affect each other. Due to the fact that there is a relationship between genes and protein products, protein interaction networks can give us a clue about genes' relationships as well.

6.2 Proposed Algorithm

GRNs are represented as graphs. A conventional method to represent a graph involves the use of a matrix where nodes are represented as rows and as columns. Cells of value 1 represent a connection; those of value 0 represent no connection. The matrix representation of a graph is not efficient for this modelling because GRNs are sparsely connected graphs. The analysis of the properties of biological networks and particularly GRNs shows us that these networks are not a complete network. They show the small world and scale-free properties. A scale-free graph is a graph in which, when the number of nodes increases, the number of edges does not increase exponentially like a complete graph. They have a few nodes with many connections and the rest of the nodes have one or two connections. For more information about the properties of GRNs see Section 9.2.

The fact that most of the nodes in a scale free graph have one or two connections and only a few nodes have many connections causes a matrix representation of such a graph to end up with a very large and sparse matrix. Such a large and sparse matrix adds to the computational overload. Furthermore, applying some operators such as *mutation* and *crossover* to add or remove an entire node or branch, or to merge two graphs, are difficult to define. Most studies related to modelling a graph represent them as a tree to make it simpler; however, this results in the loss of information. Also, there are many self-interacting elements (Prill, Iglesias et al. 2005; Alon 2007) which means there are many loops in GRNs. Considerable information will be lost if we model these networks as trees; as a result, the structure we chose was a graph.

The above reasons led us to choose subnetworks as individual solutions. We evolve these solutions to find those solutions which match the data best and then merge these subnetworks to find the complete network. We also needed to prioritize those solutions which were already known in domain knowledge to be able to generate a plausible biological output. We can advance the computational process by using existing knowledge about the dependency of genes. This knowledge is either in the form of pathways, gene functional sets or gene interaction networks. Pathways or gene interaction networks are usually represented in the bioinformatics literature as a directed graph.

The score for each subnetwork comes from two sources: one is the similarity of the subnetwork with the other biologically known graphs (or gene functional set) and the other score comes from how well they can explain the microarray data.

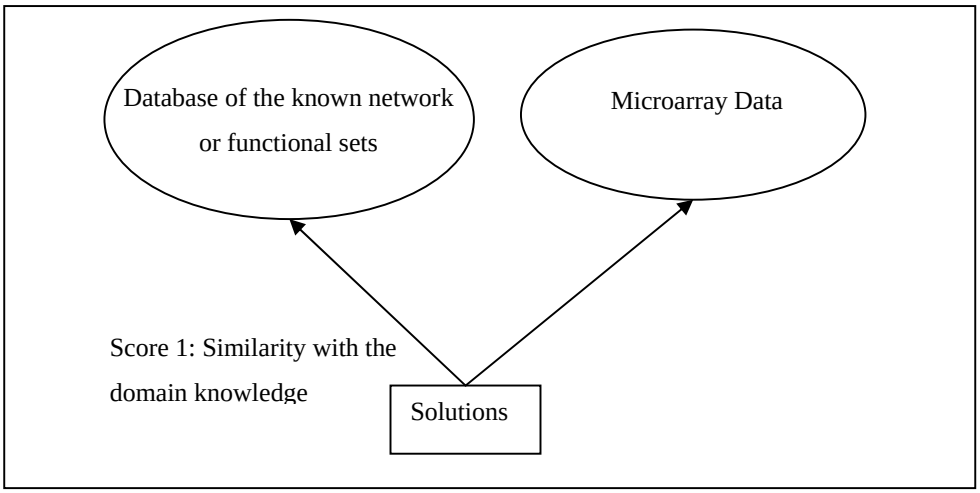


Figure 6-1 The General schematic of our scoring system for solutions in the second approach

Given that protein interaction networks and gene interaction networks are already represented as graphs, we concluded that a convenient representation is a graph. Each gene is a node in the graph and genes are connected together by edges that reflect their dependencies. As mentioned earlier, for a computational method we chose to use a Memetic Algorithm (MA) which is a combined Genetic algorithm. This decision was made for two reasons. First because GAs are known to be particularly useful for parameter estimation in complex nonlinear systems such as modelling of biological pathways (Hallinan 2008). It is also a general tool which is easy to implement, regardless of the problem under study. The second reason is that the MA gives us the facility of searching the global search space plus local improvement of the chromosomes. This property of MAs is quite compatible with the local property of GRNs. MAs have previously been used for clustering gene expressions (Merz and Zell 2002; Speer, Merz et al. 2003).

In this proposal, we used MA in a different way than in existing MAs. Instead of applying a local search method to individuals we used a case retrieval mechanism to refine and improve individuals according to domain knowledge. In other words, instead of using conventional local search methods such as *hill climbing* for improving individual solutions, we used a case retrieval mechanism to improve individuals and

make them similar to the domain knowledge. This improvement potentially has two benefits: firstly it helps to find the right answers quickly and secondly it makes the solutions more plausible. The case retrieval mechanism retrieves the most similar solutions from domain knowledge to the existing solutions found by the global search. The existing solutions then are replaced by the one found by the case retrieval mechanism. As such, we expect to find solutions which are more similar to real networks and thus more plausible. It also helps to find the solutions quicker especially in terms of finding the right structure of the graph which is hard and time- consuming by searching through the search space.

The algorithm searches for any subnetwork in the search space, and then it calculates their first score which shows how well they match the microarray data. Then the algorithm looks at the meaning of that subnetwork in terms of functional gene sets or corresponding pathways to find the second score. If there is any similarity the subnetwork gets a higher score. The similarity score is calculated based on the number of common genes between a given solution and a subnetwork of genes or a subset of genes from domain knowledge if they share more than a prespecified number of genes. The procedure of the local search (evaluation of similarity with domain knowledge) was applied only to those good solutions with a higher first score. Then we replaced the solution with the similar one from domain knowledge and we evaluated whether they score at least as well as the original one. If so, we substituted the new one. Otherwise, we left them at this stage. The procedure of local improvement can also be done by adding chains or changing existing chains in the given solution based on the biological network if it improves its ability in differentiating microarray samples.

The approach which was described above is presented in the following steps. In Chapter 4, we provided an extensive background on evolutionary algorithms but here, we give the reader a quick reminder about genetic algorithm in order to understand the following steps.

A solution in an evolutionary algorithm is represented as a chromosome. The extent to which a chromosome can solve the problem is evaluated with a fitness function so the

design of the fitness function is critical. A bunch of chromosomes evolve in order to find the best ones. The MA procedure adds an extra step inside the GA which is the local search procedure. Usually only some of the best chromosomes undergo this process.

Our memetic algorithm for GRN discovery:

- **Step 1: Generate initial population.**

  Initialize a population of individuals where each individual is a subnetwork. The initial population is generated by including all the genes in microarray data samples. The connections are generated randomly.

- **Step 2: Determine fitness of each individual**.

  This involves evaluating each individual graph against microarray data to determine how good they are in explaining the data. The fitness function is a combination of different measures. The most important one is the measure of how the subnetwork can fit to microarray data. More details about the fitness function and combination operator will be discussed in the following sections.

- **Step 3: Global Search for generating a new population.**

  Explore the search space with the GA and its exploration and exploitation operators.

  - **Apply crossover on selected sub-networks** (see more details in the next section).
  - **Apply mutation on selected sub-networks** (see more details in the next section).
  - **Fitness evaluation and selection for local search** Evaluate the fitness of each individual and select some of them to go through the improvement process using local search.

- **Step 4: Local search.**

  We will improve the best individuals or sub-networks with the procedure of extracting the similar connections from the existing networks and pathways from the literature. We substitute those similar ones with the existing ones which are found by our search algorithm if the one from the literature is scored at least as good as the existing ones. Otherwise, if we cannot find any similar solution making an improvement, we use another form of improvement by adding or changing a chain to the existing subnetwork. The chains are extracted from existing known pathways or networks.

- **Step 5: Generate new population.**

  We find the final score of each individual solution (repeat step 2) and choose the best one with a tournament selection process to generate a new generation.

Steps 3 to 5 are repeated several times to converge to the point where no significant improvement can be made or when we pass a specific number of generations.

- **Post-processing step: Merging the sub-networks.**

  After finishing the above phases we merge subgraphs together in order to build a graph which shows an overall picture. We can also map the subgraphs to corresponding pathways, as this way we can obtain more insight into the development of the condition under study. It is important to note that it is impossible to do this for all the subnetworks, as knowledge about pathways and gene functionality has not been completed yet.

We also tried different alternatives to this algorithm. One alternative was using a non random initialization. Instead of using random initialization we used pairs from known networks to initialize the solutions. The above algorithm is presented in Figure 6-2 .

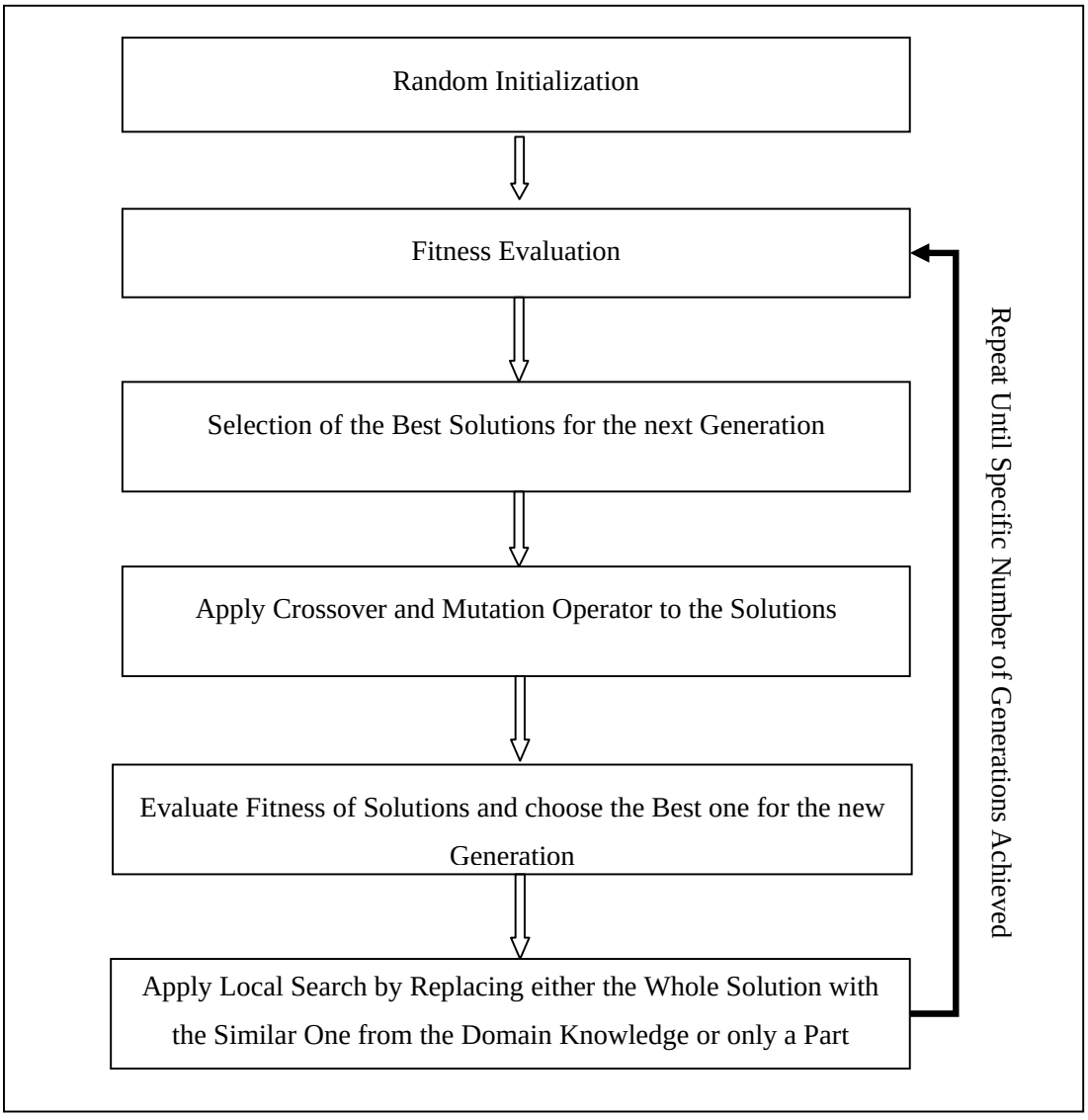


Figure 6-2 The steps of the proposed memetic algorithm

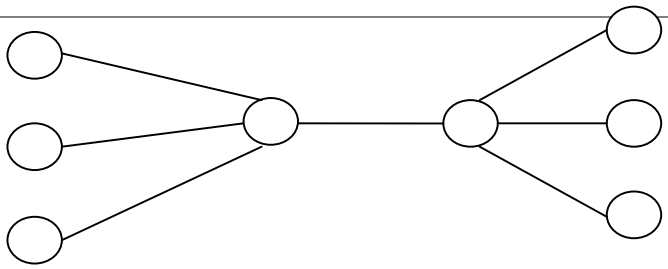
.In the above, we described the general algorithm. In the following section we will explain the detailed design and assumptions that underpin the overall presented picture. We will start with describing how we determined the first score of solutions. This score was an indicator of similarity of solutions with the domain knowledge. Subsequently, we

will describe the second score for the solutions which is the fitness function. Finally, we will describe variation operators in global search and the mechanism for the local search process.

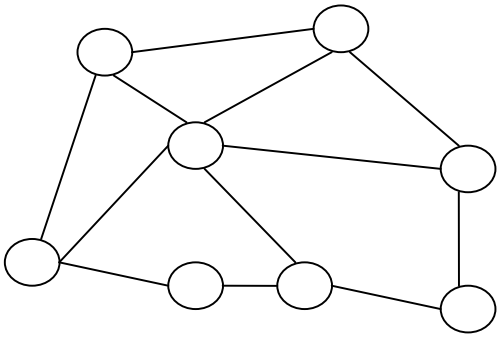
6.3 Domain Knowledge for Scoring the Solutions

We need some criteria to evaluate the similarity of our solutions with the solutions from the domain knowledge as it is important to produce a plausible result. This can also guide the search process to find the solutions more effectively. In order for us to find the properties and shape of the known GRN we did a comprehensive literature review, which the reader can find in Section 9.2 where we will review it and use it again in our third approach. Here we only review two properties that we used in this algorithm. Our first measure is an indicator of the shape of the solution and the second one recognizes difference in the total distance of the genes in the subnetwork compared with the corresponding biological network. The criteria that we proposed in this study are as follows:

- Biological Networks are known to be organized through hubs. Hubs are those nodes which have a large number of connections and are known to be the most important nodes in the network (for more information read Section 9.2). Therefore, we should prefer to select those solutions (sub-networks) that have nodes organized through hubs. For example, in Figure 6-3 we preferred the architecture uppermost than the lower most.



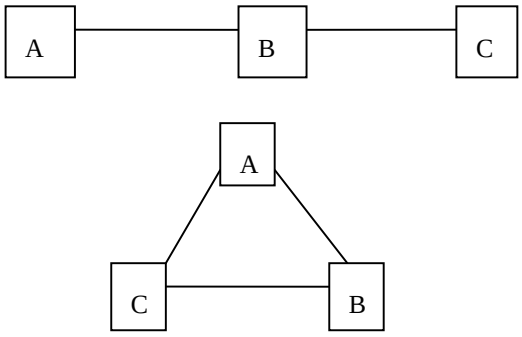
a) An example of a fit subnetwork



b) An example of an unfit subnetwork

Figure 6-3 Shows samples of desired (a) and undesired (b) subnetwork solutions

- It is necessary to compare the proximity of nodes in a solution with their proximity in known biological networks. For this, we needed to define a measure of the distance between two nodes in a graph. For the fitness function to be used in this study we considered that two nodes are *x* units apart if there are *x* other nodes between them. Adjacent nodes are scored "1". For example, consider AC in the upper part of Figure 6-4. The distance between nodes A and C is "0" because there are no connections between them and they are not adjacent.



$$d_{ab} = 1$$
$$d_{bc} = 1$$
$$d_{ac} = 0$$

$$d_{ab} = 1$$
$$d_{bc} = 1$$
$$d_{ac} = 1$$

Figure 6-4 Sample of distance calculation

Consider two nodes *n1*, *n2* for which a typical solution can be as follows:

$$\frac{\sum_{i,j=1}^{n}(\text{SDistance } ij - \text{Distance } ij)^2}{N^2}$$

Where:

a. SDistance is the distance between nodes *i* and *j* in the solution

b. Distance is the distance between nodes *i* and *j* in the domain knowledge

c. *N* is the total number of nodes in the sub-network

These two criteria can be used to find the similarity of our solutions to the similar subgraphs in the domain knowledge. We considered the first criteria in our penalty function and we used the second criteria as the local search.

6.4 Fitness Function

We previously mentioned that there are two measures for evaluating a good solution: one is the score which comes from evaluating the similarity of the solution to the domain knowledge and the other one is the fitness function which tells us how well a solution

can explain the microarray data. In this section we will describe our second score which indicates how well a solution matches microarray data. This score is used as the fitness function.

Our fitness function measures how a set of features (genes) are dependent on each other with regards to the microarray data. Among the measurements of the independence between random variables, mutual information is singled out. It is sensitive to any dependencies which do not show up by covariance. Mutual information is "0" only if two variables are not dependent at all.

Fitness function is a central part of a GA; therefore, we will describe the Mutual Information here by providing an example. The approach of applying Mutual Information to the subnetworks was adopted from the scoring protein network interaction introduced by Chuang et al. (2007).

Expected Mutual Information:

Here we provide an example of calculating Expected Mutual Information. In order for the reader to understand this we provide a simple example. In this example there are two class labels, "diseased" and "normal". Given a particular gene set M, let a represent its vector of activity scores over the diseased sample and let c represent the corresponding vector of class labels. To calculate a, expression values $g_{i,j}$ are normalized to z-transformed scores $z_{i,j}$ which for each gene i has a mean $\mu = 0$ and the standard deviation $\sigma = 1$ over all samples J. The individual $z_{i,j}$ of each member gene in the gene set are averaged into a combined z-score, which is designated the activity a_j. Example 1 illustrates the Expected Mutual Information (EMI).

Example 1.

Original activity: a = [-2 -1 1 2 -4 -2 2 4] where scores are gene expression levels for individuals.

Discretized activity: $\acute{a}$= [-1.15 -1.15 1.15 1.15 -3.45 -1.15 1.15 3.45] where scores are z-transformed transformations of a.

Class Label: C= [1 1 1 1 2 2 2 2] where scores are class labels 1= diseased and 2= non-diseased.

		x				
		-3.45	-1.15	1.15	3.45	
C	1	0	1/4	1/4	0	1/2
	2	1/8	1/8	1/8	1/8	1/2
		1/8	3/8	3/8	1/8	

Figure 6-5 Mutual Information between activities and class labels

$$MI(a,c) = \sum_{x}\sum_{y} p(x,y)\log\frac{P(x,y)}{P(x)P(y)} = \left(\frac{1}{4} * \log\left(\frac{\frac{1}{4}}{\frac{3}{8}*\frac{1}{2}}\right)\right) * 2 + \left(\frac{1}{8} * \log\left(\frac{\frac{1}{8}}{\frac{1}{8}*\frac{1}{2}}\right)\right) * 2 + \left(\frac{1}{8} * \log\left(\frac{\frac{1}{8}}{\frac{3}{8}*\frac{1}{2}}\right)\right) * 2 = 0.215 \quad (6.1)$$

Equation (6.1) depicts the Expected Mutual Information as described by Chuang et al. (2007) between *a* and *c*.

Example 2.

A more general case can be considered when we need to calculate MI between a pair of genes. The following example shows the calculation of MI for a gene pair.

Consider discretize profile for two genes A and B are: A=[1,1,0,1,-1] and B=[1,-1,0,1,-1]. The probability for each combination to occur is as follows:

	Probability			
Genes	P(1)	P(0)	P(-1)	P(1)+P(0)+P(-1)
A	3/5	1/5	1/5	(3+1+1)/5=1
B	2/5	1/5	2/5	(2+1+2)/5=1

Figure 6-6 Step 1 in calculation of Mutual Information between two genes- calculation of individual probabilities

From the above table the Shanon's etnropy for each gene can be calculated as follows:

$$H(gene) = \sum_{i=1}^{3} P * \ log_p i \tag{6.2}$$

$$H(A) = -1 * \left(\frac{3}{5} * log_2 \frac{3}{5} + \frac{1}{5} * \ log_2 \frac{1}{5} + \frac{1}{5} * \ log_2 \frac{1}{5}\right) = 1.371$$

$$H(B) = -1 * \left(\frac{2}{5} * log_2 \frac{2}{5} + \frac{1}{5} * \ log_2 \frac{1}{5} + \frac{2}{5} * \ log_2 \frac{2}{5}\right) = 1.52$$

Now, the third step is calculation of pairwise combination of states.

P(A,B)	Frequency of occurrence
P(1,1)	2/5
P(1,0)	0/5
P(1,-1)	1/5
..	..

Figure 6-7 Step 3 in calculation of Mutual Information between two genes- frequency of pairwise combination

And then in the fourth step, we need to calculate the joint entropy $H(A,B)$.

$$H(A,B) = \sum_{i,j=1}^{3} P_{i,j} * log_2 \, \mathrm{P_{i,j}} = -1 * (\tfrac{2}{5} * log_2 \tfrac{1}{5} + \tfrac{1}{5} * log_2 \tfrac{1}{5} + \tfrac{1}{5} * log_2 \tfrac{1}{5}) = 1.923$$

Finally Mutual information between expression profiles of two genes is calculated.

$$M(A,B) = H(A) + H(B) - H(A,B) = 1.371 + 1.522 - 1.923 = 0.97$$

In addition to the Expected Mutual Information, we proposed a penalty function that takes into account the requirements that we discussed earlier in section 6.3.

6.5 Penalty Function

The Penalty function will be determined using the same formula for the topological measurement of protein interactions by Pei and Zhang (2005) which has resulted in the measure that was called the *cluster coefficient.* A clustering coefficient is defined as the edge density around a node's neighbours. In a small-world protein interaction network, a high clustering coefficient property predicates that proteins are likely to form dense clusters by interactions. We chose to measure the significance of two proteins co-existing in a dense network as an indication of interaction reliability. In Equation (6.3), *N(A)* and *N(B)* means show the number of neighbours for node *A* and node *B*.

$$\frac{|N(A) \cap N(B)|}{\sqrt{|N(A)| * |N(B)|}} \tag{6.3}$$

The penalty function is applied at the stage when chromosomes are selected to undergo the local search process.

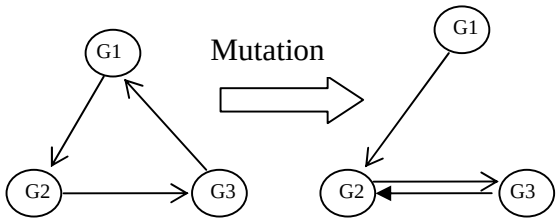
In general, we use the following three criteria in this algorithm to evaluate a score for each individual solution.

- Fitness Function: Expected Mutual Information (or alternatively Chi-Square) for the estimation of the dependency of a set of genes considering our microarray data. This metric has been used in microarray studies by Chuang et al. (2007) to distinguish the microarray data from diseased and non-diseased samples.
- Local Search: Difference in the total distance of our solution with corresponding genes in the other gene regulatory networks or protein interaction networks.
- Penalty Function: A penalty function can be considered for solutions having genes with fewer edges.

6.6 Combination Operators

Parameters for the settings of the global search component of the MA were drawn from a study by Speer (2003) and the GA literature. An initial population of 70 chromosomes over 200 generations with a probability of crossover 0.5, and probability of mutation 0.07 was initially considered. Further tests and tuning were done to determine the parameters according to our algorithm. Crossover and mutation operators for individuals which are represented as a graph cannot easily be drawn from other studies, because few studies have explored the graph representation. Inspired by the combination operators of the study by Mabu et al. (2007) which also used the graph representation of chromosomes we designed the following combination operators. However, in that work a graph represents a complete solution.

Mutation: For mutation, we defined an operator to change the connections randomly based on the mutation rate. In order for the reader to understand this operation, Figure 6-8 is provided. We chose an individual based on the mutation probability, and then we chose one of its branches by another probability. We then removed it and randomly connected that node to the other node to create another branch. The rate and probability of the mutation operator is arbitrary and is usually chosen by experience.



Each branch is selected with the probability of Pm (Probability of mutation)

The selected branch becomes connected to another node randomly

Figure 6-8 Mutation operator

Crossover:

Crossover takes two individual solutions (parents) and generates two offspring. The procedure of crossover is as follows:

1. We selected one individual using tournament selection then selected the second individual based on the tournament selection biased towards the fittest individual that had genes in common with the first one. In this way, we biased our selection towards finding individuals with common genes for use as parents.
2. We preferred to find the common nodes (biological gene) to perform crossover operations on them. Therefore, we chose a set of nodes which were common between individuals. Then we selected one node based on user configured probability of Pc (Probability of Crossover). If two individuals did not have any node in common, a node *i* was selected as a crossover node with the probability of Pc. Two parents exchanged the edges of the selected node for crossover. Figure 6-9 illustrates the crossover operation.

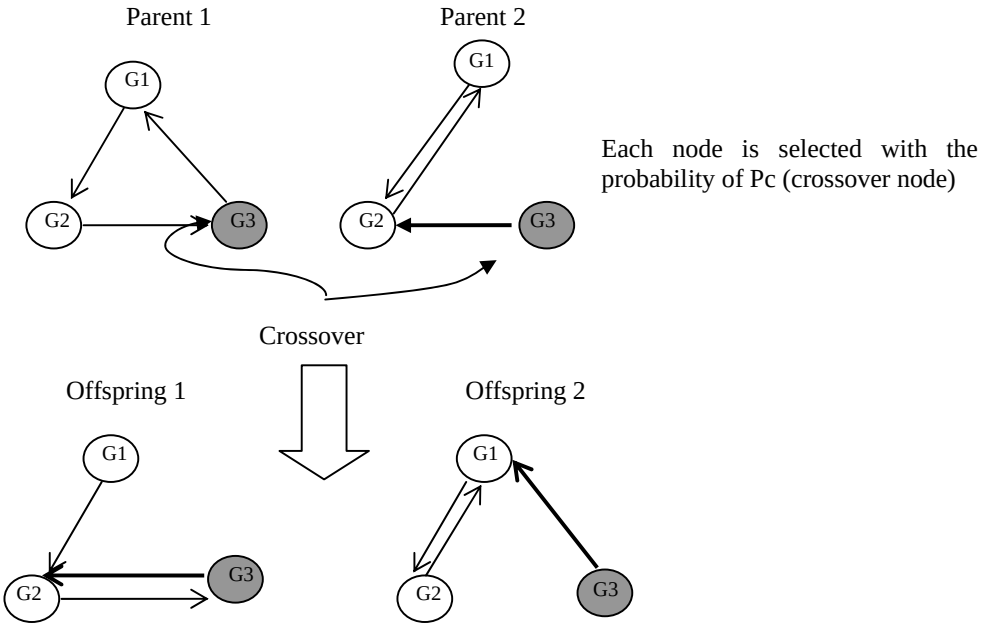


Figure 6-9 Crossover operator

Selection: The selection process which we used in our algorithm was tournament selection. Figure 6-10 represents the code for the tournament selection which we implemented. Each time, based on the tournament pressure a portion of population was selected randomly and the best of them were added to the next generation. This procedure continued until we finished with a population as large as the previous one.

```
def tournamentSelection(self):
    nextGeneration = Genetics() \\ Initialize population of solutions
    for chromo in self.curGen: \\For each chromosome in population
        chromo.fitnessCalculation() \\Calculate Fitness function
    \\ do this operation until we reach to the same size population again
    while len(nextGeneration) < POPULATION:
        \\take a random sample from the population to generate the new generation
        tournament = random.sample (self.curGen,TOURNAMENT_P)
        tournament.sort (reverse=True) \\sort the sample pool
        \\copy chromosomes from the sample pool to make a new generation
        nextGeneration.append (copy.deepcopy(tournament[0]))
return nextGeneration
```

Figure 6-10 Tournament selection code

6.7 Local Search

In memetic algorithms local search usually plays the role of local improvement and is implemented with a heuristic or exact method. We had a GA which searched the global search space in order to find the possible subsets of genes which were able to differentiate between two groups of samples in microarray data. On the other hand we needed solutions that match with the existing domain knowledge. Related domain knowledge can be in the form of GRNs, pathways, Gene Ontology sets and protein network interactions (for the definition of these terms see the glossary). All of these different sources can give us a clue about any association between genes. It has been demonstrated that using integrated information sources will improve the GRN discovery process (Zhu, Zhang et al. 2008).

This implies that we will improve the process if we can improve the subnetworks which we found by our global search process. We did this by making the solutions similar to the corresponding subnetworks in the domain knowledge. We implemented this idea as follows: we performed a global search to find the possible solutions according to microarray data but at the same time we wanted to have solutions compatible with biological knowledge. Therefore, we improved the solutions by a local search process.

The local search process in previous MAs was usually a neighbourhood search which improved the individual solutions. Here we defined a different local search process. Our local search process looked at the information from the domain knowledge in order to change and improve our solutions by making them similar to the domain knowledge. We called this part *imitation from culture.*

The above local search was implemented in two ways. In the first way, given a subnetwork, we looked for the similar subnetworks in domain knowledge. If a subnetwork was found, then we substituted the new network; otherwise, if an exact match was not found, chains from the best matched subnetwork were added to or removed from the existing subnetwork.

6.7.1 Balance between Genetic and Local Search

An issue that needs to be considered in the design of a memetic algorithm is how to establish the right switching mechanism between local search and genetic search. Ideally, operators that belong to these groups should work together in cooperation instead of against each other (Burke and Landa-Silva 2004). Burke and Landa-Silva also mentioned that for making a right balance between genetic and local search, three questions need to be addressed:

- What is the right balance between the local search and genetic search operators?
- Which solutions undergo the local search?
- What is the best balance between the local search and the global search in terms of computing time?

The following answers to the above questions were extracted from (Krasnogor & Smith, 2005). It was shown that even within a single problem class (in that case Travelling Salesman Problem) the choice of a single LS operator which gives the best results when incorporated in an MA is entirely instance specific. Furthermore, the studies of the dynamic behaviour of various algorithms showed that in fact the choice of the local search operator yielded the biggest improvements and was also time-dependent.

Therefore we needed to find the balance between local and global searches at run time by tuning the algorithm as it depends on the distribution of individuals in the population (El-Mihoub, Hopgood et al. 2006). The answer for the second question is to choose only the best individuals (probably only 20%) to apply a local search to them. Of course, these are only the starting points and these parameters have to be tuned in practice by testing different values in each run of the algorithm. The best balance between GA and local search can only be achieved at runtime, via an experimental process.

```
def improve(self, percent):
    global knownPairs
    replaceNum = int(self.size() * percent)
    self.genes.sort(reverse = True)
    self.genes = self.genes[0:-replaceNum]
    availablePairs = []
    for pair1 in knownPairs:
        add =True
        for pair2 in self.genes:
            if pair1.genepair==pair2.genepair:
                add = False
                break
            if add:
                availablePairs.append(pair1)
    if (replaceNum>len(availablePairs):
        temp1=replaceNum%len(availablePairs)
        temp2=len(availablePairs)/2
        self.genes.extend(random.sample(availablePairs, temp2))
        self.genes.extend(random.sample(availablePairs, temp2))
        self.genes.extend(random.sample(availablePairs, temp1))
    else:

       self.genes.extend(random.sample(availablePairs,replaceNum))
    random.shuffle(self.genes)
```

Figure 6-11 Our local search algorithm

6.8 Data and Tools

D**ata:** A computational technique for GRN inference can be tested by measuring its performance on real test beds (known networks) or on synthetic data. It is obvious that

having a real test bed is a better way of measuring a performance of algorithms but for so many reasons as mentioned earlier in Chapter 3 (Experimental Setup), this is not applicable yet. Therefore, for this study we used simulated data which was produced using a gene regulatory network simulator called SynTReN. The first dataset described in the experimental chapter was used to test this algorithm. The parameters and settings to generate the first dataset were described in Section 3.6.

Tools: In this study we used SynTReN to generate the datasets and we used Cytoscape to visualize the network and also to retrieve the related information for subnetworks and genes by using its several integrated tools.

Another important tool that we used for the retrieval of similar cases in CBR module was BABELOMICS (Al-Shahrour, Minguez et al. 2006). BABELOMICS is a suitable tool for this purpose as it uses several different molecular biology resources to extract information including Go, KEGG pathways and Gene expression in tissues.

A Cytoscape plug-in, ActiveModules, which can find a subnetwork for each condition can also be used (Shannon, Markiel et al. 2003). This tool was previously used in a network enrichment analysis in the study by Chuang et al. (2007).

6.9 Results of the Proposed Memetic Algorithm

In this experiment, we tried different sizes for chromosomes, different generations, different crossover rates, and different mutation rates. The best results were achieved with the largest chromosome size and only 30 generations. The algorithm converged after 27 generations and we could not find any solution for the early convergence problem, despite trying different mutation and crossover rates.

Table 6-1 Result of our Memetic Algorithm approach

Number of Generations	Chromosome Size	True Positives	False Positives	Precision	Recall	F-measure
30	30	12	544	0.0216	0.041	0.0282
30	50	25	940	0.026	0.085	0.0398
30	70	34	1329	0.025	0.116	0.041

As shown in Table 6-1, as the chromosome size increases, the number of true positives increases therefore the performance increases. We could not try a chromosome with the size larger than 70 genes (genes here are evolutionary units not biological genes) because of a memory overflow problem. However, it was obvious from the result that we would not achieve a good performance anyway, as the number of false positives grows as well as the number of true positives. Despite the F-measure being improved by increasing the size of chromosomes, the result was still far from the best reported performance in the literature with an F-measure 0.1 previously reported in a similar situation (Leemput, Bulcke et al. 2008). The above result was obtained by using only our GA and we did not add the local search. The reason behind not using the local search was that GA itself did not work well and suffered from early convergence. Local search usually causes faster convergence and by using it we only made the situation worse.

This experiment proved that GA is not a suitable technique for this type of complicated search space. We also found that the design of GA to work on partial solutions is still in its infancy (Yang, Tang et al. 2008) based on which no strong algorithm can be established.

6.10 Summary

In this chapter, we proposed our second approach which used a type of evolutionary process to find the subnetworks and a different local search process to make the solutions similar to the domain knowledge. Our attempt at designing a combined evolutionary algorithm using information from domain knowledge showed us that GA works well when the search space is large and coding is simple. However in this application, we could not use the simple GA to find the GRN structure, as that required a huge chromosome which was hard to evolve in order to find the solution. That is why we tried to build the whole GRN by evolving its smallest elements (functional gene subnetworks). This also gave us a facility to incorporate the domain knowledge which is in the form of gene subsets and subnetworks. Our GA mechanism suffered from early convergence and therefore, we could not achieve a good result. We could not find a solution for the early

convergence problem and we concluded that GA is not practical in such a search space. Using partial chromosomes in GA has not been studied yet. This area is still in an early stage of development and it takes more time and study before such a mechanism can be applied to a real application.

We realized from this experiment that the search space is too large and complex to be explored by a GA. The solution which we found to limit the search space was using more information from the domain knowledge to limit the search space from the outset. In the next chapter we will describe our third approach which used more heuristics from the domain knowledge from the outset in order to limit the search space. The third approach also questioned the usefulness of the dependency measure which we used here as a fitness function (MI) and arrived at a new association measure.

Chapter 7

Approach 3: Heuristics Based on Regulatory Relationships

"It is the theory which decides what we can observe"

— Albert Einstein

7.1 An Overview of Approach 3

In the previous two chapters, we described two approaches toward the GRN discovery problem. We reviewed their weaknesses and limitations which gave us a basis to develop a new approach introduced in this chapter. In the previous chapter (Chapter 6), we reviewed an approach which used a combined evolutionary algorithm for the GRN discovery problem. That approach considered some information related to the properties of GRNs to design a customized GA for solving the problem. We reported that the GA mechanism did not work well because the design of an evolutionary algorithm in such a way made it a complicated algorithm and prone to early convergence. We also did not have enough freedom to use as much information inside the evolutionary algorithm.

In this chapter and the two following chapters, we will introduce our third approach which uses more information from domain knowledge directly. In doing that, we changed our technique as we found in the second approach; GA is not a suitable technique for this purpose. GA is usually effective when there is a huge search space with simple encoding. By using domain knowledge we ended with a complicated encoding that made it difficult for a GA to converge. Moreover, we do not really need GA as the search space is not that large if we use domain knowledge from the outset to limit the search space.

That is why in the third approach, we considered more information from the domain knowledge to solve the problem as our previous attempts had demonstrated that we needed to use more domain knowledge. There is also much evidence that confirms using more information from domain knowledge will improve the process considerably (Zhu, Zhang et al. 2008).

The main research question of this thesis was *"How can reliance on microarray data and heuristics be reconciled to improve GRN discovery?"* In this approach, we answered this question by using the maximum amount of heuristics in several ways and comparing the results with methods that do not use such information. The third approach followed two strands. In the first strand, we used the definitions of regulatory relationships in order to design an association function for measuring pairwise dependencies between genes more accurately and plausibly. The proposed association function is called a *2D Visualized Co-regulation Function* and has great visualization ability. In the second strand, we used the structural properties of the known GRNs to design an effective algorithm for GRN discovery.

In this chapter we will explain the theory behind our co-regulation function. First in section 7.2 we will review some of the well-known association measures and in section 7.3 discuss the limitations of each of them. In section 7.4, we will then review the desirable properties for an association measure for measuring gene-gene relationships. These two sections answer the question of *"How to explain observed gene expression data in terms of co-regulation rather than correlation?"* Following on in section 7.5, we will propose our new association measure to answer the question of *"How can we measure the association between two genes more precisely compared with the existing functions?"*

In Chapter 8, we will present the experiments related to our proposed co-regulation function. Chapter 9 will review the second strand of the third approach which employed the structural properties of the known GRN to design an effective algorithm. A comprehensive literature review and analysis was carried out to specify the gene regulatory network properties. We identified some heuristics based on this analysis and

used them in order to find the right structure of the target GRN more effectively. The question that we will answer in that chapter will be *"How can we use the properties of known gene regulatory network (such as structural properties) in order to design a more effective discovery algorithm?"*

7.2 Association Measures

Association measures are mathematical formulas which interpret co-occurrence frequency data. For the construction of gene regulatory networks in the form of correlation networks the pairwise associations between genes has to be measured.

Such a measure specifies the strength of relationship between two variables. There are two types of association measures: *similarity measures* and *dissimilarity measures*. Similarity measures reflect the similarity between variables and dissimilarity measures reflect the dissimilarity of variables. A direct relationship between similarity and dissimilarity may occur but it cannot be assumed that it is always relevant.

Goodman and Kruskal (1954) support the idea that the measure of association used by an empirical investigator should not be blindly chosen because of tradition and convention only. Although these factors may properly be given some weight, but they should be constructed in a manner that has operational meaning within the context of the particular problem. They also stated that even when a single precise goal for the investigation cannot be specified it is still desirable and possible to choose a measure of association which has a contextual meaning. They emphasize the fact that the measure of association should have operationally meaningful interpretations that are relevant in the contexts of empirical investigations in which the measures are used.

In the context of gene regulatory networks (GRN), measuring the associations between genes is the first and a crucial process in constructing the network. There are several tools and functions from statistics to information theory for measuring the associations between genes. In the GRN literature some of the most commonly used ones are correlation coefficient (Pearson's correlation) and expected Mutual Information (MI).

There is a limited research exploring association measures for GRNs. Little is reported on different association measures, their comparative performance and justifications for the use of one over another. Therefore this chapter will review some of the common measures for GRN and will provide desirable characteristics for an association measure for the GRN discovery purpose. Finally, we will propose a new association measure that has most of those desirable characteristics.

7.3 Comparing Association Measures for GRN Discovery

Here, we will review some of the most common association measures for GRN discovery. The list of association measures for this purpose is long but we do not consider all of them here. Instead we will focus on the common ones that have been reported to perform better than the others. For a better understanding of how different association measures work, we will provide examples by using a toy dataset provided in Table 7-1.

The toy dataset is produced by using a synthetic data generator called SynTReN. In this dataset there are six genes and 20 samples. We will explore how different association measures work on this dataset, what type of assumptions they follow and what are their limitations.

Table 7-1 A toy dataset

	G1	G2	G3	G4	G5	G6
sample_0	0	0.1	0.1	0.1	0	0.9
sample_1	0.1	0	0.3	0.4	0.3	1
sample_2	0.2	0.3	0.4	0.6	0.1	0.9
sample_3	0.2	0.4	0.5	0.3	0.5	0.8
sample_4	0.2	0.3	0.5	0.6	0.4	0.7
sample_5	0.2	0.7	0.6	0.8	0.2	0.1
sample_6	0.3	0.1	0.6	1	0.2	0.7
sample_7	0.3	0.1	0.7	0.2	0.3	0.8
sample_8	0.4	0.3	0.7	0.6	0.6	1
sample_9	0.4	0.4	0.7	0.8	0.1	0.7
sample_10	0.4	0.3	0.8	0.8	0.4	0.9
sample_11	0.4	0.6	0.8	0.7	0.3	0.5
sample_12	0.5	0.6	0.8	0.9	0.1	0.7
sample_13	0.5	0.4	0.8	0.9	0.2	0.6
sample_14	0.6	0.2	0.9	0.5	0.8	0.5
sample_15	0.6	1	0.9	0.7	0.7	0.3
sample_16	0.7	0.5	0.9	0.8	0.6	0.9
sample_17	0.7	0.8	0.9	0.9	0.3	0.9
sample_18	0.8	0.9	1	0.9	0.4	0.9
sample_19	0.9	0.8	1	0.9	0.9	0.7

7.3.1 Correlation Coefficient

The correlation coefficient belongs to the group of similarity measures and indicates the strength of a linear relationship between two variables. The most used correlation measure is Pearson's correlation (PC). Pearson's correlation describes the linear relationship between two variables. Its values range from -1 to 1. The formula in Equation (7.1) calculates Pearson's correlation (Soranzo, Bianconi et al. 2007) between variable *x* and *y*.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})\,(y_i - \bar{y})}{(n-1)S_x S_y} \tag{7.1}$$

Where n is the number of observation, S_x is the standard deviation of x, and S_y is the standard deviation of y. For example, Pearson's correlation between G1=x and G2=y is calculated as follows:

$\bar{x} = 0.42 \;\; \bar{y} = 0.44$

$r = \frac{0.914}{\sqrt{1.59 * 1.11}} = 0.69$

Examples in GRN Literature:

Pearson's correlation is one of the first measures used in the literature of GRN for measuring pairwise dependencies. An early example of this appears in a work by Butte and Kohane (1999). They calculated pairwise Pearson's correlation to build a relevance network. Pearson's correlation also was used in (Zhu, Zhang et al. 2008) to build a weighted co-expression network. In a study by Zhou and et al. (2005) first order Pearson's correlation (the correlation between each pair) and the second order correlation (the correlation between the first order items) were used to build the network.

A more precise way of measuring the correlation between two variables is considering the partial correlation. A partial Pearson's correlation measures the correlation between two genes conditioned on one or several other genes. The number of genes conditioning the correlation determines the order of the partial correlation. In a package called ParCorA by de la Fuente et al.(2004) the partial correlations of up to 3rd order were implemented.

Assumptions, Strengths and Limitations:

- Pearson's correlation assumes that data is normally distributed and therefore is not skewed either negatively or positively.
- Pearson's correlation can be applied on real values as well as discrete values.
- Pearson's correlation looks for linear relationships and is not able to detect any relationship which does not appear in any of two diagonals when we draw a two dimensional histogram from two genes. It has poor visualization ability as it reduces the relationship to a single number.

7.3.2 Information Theory

Inspired by the increasing power of computation and the successful application of information theory in telecommunication and physics, similarity measures from the information theory have recently been used for GRN inference (Margolin, Nemenman et al. 2006; Watkinson, Liang et al. 2009). Expected mutual information is a commonly used one. It is a general measure of the dependency between two variables. Mutual information measures how much more is known about one random value when we know the value of another. For example, by knowing that a person has a high blood pressure we can better predict the risk of a heart attack. In other words, mutual information measures the difference in predictability when considering two variables together versus considering them independently Mutual information can be computed in different ways, but often is based on Shannon's entropy formula which is shown in (7.2):

$$I(Xx, Y) = H(X) + H(Y) - H(X, Y) \tag{7.2}$$

$I(X, Y)$ is the amount of information between X and Y. $H(X)$ is the entropy or the amount of uncertainty of variable X. $H(X|Y)$ is the conditional entropy of two random variables x and y and is a measure of the uncertainty in X once we know Y.

Entropy can be considered as the measure of probability distribution and can be formulated as follows:

$$\mathrm{H(X)} = -\sum_{x \in \Omega_x} \mathrm{P(X = x)} \log \mathrm{P(X = x)} \tag{7.3}$$

In such a definition then mutual information is a measure of the difference between the joint probability and product of the individual probabilities. These two distributions are equivalent only when x and y are independent, and diverge as x and y become more dependent. Mutual information can be considered as the difference between two distributions and is equal to:

$$I\,(X,Y) = \sum_{x\in X}\sum_{y\in Y} P(x,y)\log\left(\frac{P(x,y)}{P(x)P(y)}\right) \tag{7.4}$$

Where I is the mutual information between two variables X and Y. $P(x,y)$is the joint probability distribution of x and y. $P(x)$ and $P(y)$ are the marginal probability distribution functions of x and y.

This formula works only on discrete values not on real values; therefore, microarray data needs to be discretized. Here for simplicity our toy dataset has ten different values between 0 and 1 and we discretized it in 5 categories, as shown in Table 7-2 for genes G1 and G2:

Table 7-2 Example of Mutual Information calculation

G1 \ G2	0.1-0.2	0.3-0.4	0.5-0.6	0.7-0.8	0.9-1	*Total*
0.1-0.2	2/20	3/20	0	1/20	0	*6/20*
0.3-0.4	2/20	3/20	1/20	0	0	*6/20*
0.5-0.6	2/20	1/20	1/20	0	1/20	*5/20*
0.7-0.8	0	0	1/20	1/20	1/20	*3/20*
0.9-1	0	0	0	1/20	0	*1/20*
Total	*6/20*	*7/20*	*3/20*	*3/20*	*2/20*	

$$\text{MI}= \sum_{x\in X}\ \sum_{y\in Y} P(x,y)\log\left(\frac{P(x,y)}{P(x)P(y)}\right) = 2 * \frac{1}{10} * \log\left(\frac{1}{10}/\left(\frac{1}{4}*\frac{1}{10}\right)\right)=0.3 \tag{7.5}$$

Examples in GRN Literature:

Mutual information has been widely used in the literature for GRN discovery. Recent studies tend to use mutual information as the measure of dependency of two variables. Some of the best-performing packages developed for GRN discovery use mutual information. For example, the famous study called ARACNE (Margolin, Nemenman et al. 2006) used mutual information along with Data Processing Inequality (DPI) and

reported good performance. DPI is a method to distinguish between direct and indirect relationships. In each triplet of fully connected nodes in the network obtained after applying mutual information, the edges with the lowest mutual information are removed as these edges are indicators of indirect relationships. Another example of using mutual information is CLR which uses mutual information along with background correction to eliminate indirect influence (Chuang, Lee et al. 2007) and another recent example is (Watkinson, Liang et al. 2009).

Assumptions, Strengths and limitations:

- Mutual information does not assume a linear relationship between any given pair of genes (or any parametric relationship, for that matter). Therefore, it is more general than Pearson's correlation as it tells us about differences not only in the mean but also in variance of expressions of two genes.
- It does not require a normal distribution.
- MI scales well to genome-wide regulatory networks, where the functional forms of regulatory interactions are unknown, complex, or where there is insufficient data to learn more intricate models (Madar, Greenfield et al. 2010).
- MI based methods provide limited insight into the dynamic behaviour of the system, and hence have limited use in predicting new observations—a key property for estimating a model's relevance when the ground truth is unknown.
- MI does not work particularly well when two variables are low together at the same time. Also, there are some other conditions where MI fails to detect the pattern, such as the case presented in Figure 7-1.
- Like Pearson's correlation MI has limited potential for the visualization of gene relationships.

Figure 7-1 presents an example where MI fails to detect a pattern as the MI score is 0.34. This is lower than the 0.5 threshold in the scale of 0 to 1. As mentioned, MI also does not work as effectively when a pattern exists in the Low-Low area because the information gain reduces.

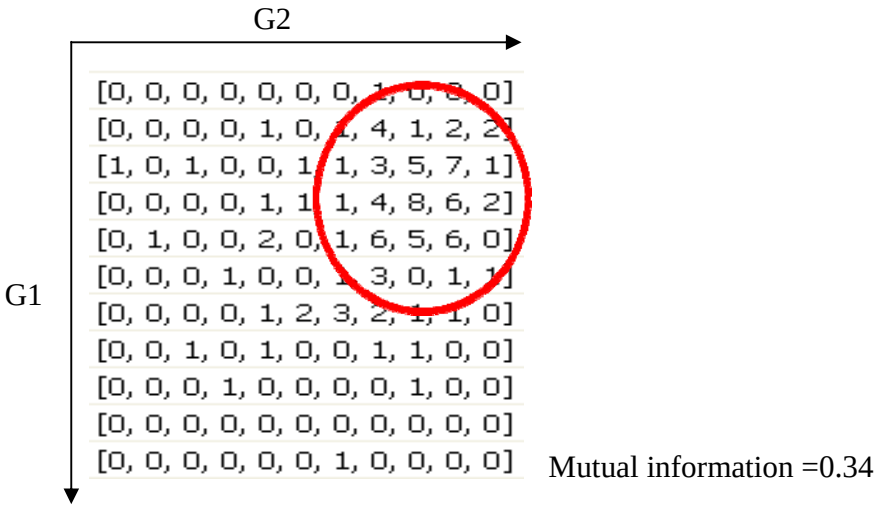


Figure 7-1 Examples where MI fails to detect a pattern

7.4 Desirable Association Measure for Gene Co-expression

We reviewed two widely used association measures in the past sections. We described pros and cons of each measure. Based on this review so far none of the association measures look exactly for the regulatory relationships that we are going to detect in GRN discovery. There is a lack of association measures based on the definitions of regulatory relationships. Measures reviewed so far detect general dependencies (MI) or a special kind of dependencies such as the linear dependency (PC).

The first question here is "What is the nature of the regulatory relationships that we are going to discover?" We need a function to measure the regulatory relationship specifically. Another important characteristic for an association function is having plausible results. Visualization can help the expert to get a sense of what is the exact pattern of the relationship between two genes. Producing a number to explain a relationship is not informative enough. For example, MI is able to produce only a positive number which does not tell anything about the nature of the relationship (whether it is activation or suppression).

The function also needs to have the ability to scale up and it is important to be simple to implement. The following is a summary of characteristics that we expect an association function to exhibit:

- Allow for visualization

  Visualization is a key concept of understanding and analysing the expression data. It makes the result meaningful and plausible and gives biological insight (Gehlenborg, O'Donoghue et al. 2010) .
- No data assumptions

  The nature of data is so complicated and full of variations therefore, the method should not consider any particular distribution or assumption.
- Accurate

  Accuracy of current methods on this application is low and unsatisfactory (Marbach, Mattiussi et al. 2009). Therefore, it is important to elevate the performance.
- Easy to implement.

  This is a relatively minor consideration but still makes some difference in practice.
- Scale up

  It is important for a measure of association to scale up, especially in reality, where the size of networks is large; therefore, this is an important characteristic. For example, when we apply an association measure on a protein interaction network the size of the network is huge. Thus a tool such as Bayesian Networks cannot be applied.

7.5 Co-regulation Measure Based on Heuristics (Visualized Co-regulation Function)

Considering the above characteristics list, we designed our co-regulation function to satisfy those requirements. We defined a co-regulation function which works based on

the definition of the main regulatory relationships. The function also supports the visualization of the gene relationships. The visualization ability of the function gives an expert the opportunity to see clearly the overall behaviour of any gene pairs across the samples without any assumptions about the data.

In general there are three major regulatory relationships: up regulation, down regulation and dual interaction. Up regulation is where a product of one gene causes the other gene to express highly. This definition does not tell us anything about when the amount of the first gene is low what would be the behaviour of the second gene. The second gene could be low or high as a result of the influence of other genes. Therefore, the only conclusion about this type of the relationship, is that most of the time when the first gene is high, the second gene is high as well.

Down regulation is where a product of a gene causes another gene to be suppressed. Again, this definition does not tell us anything about when the amount of the first gene is low what would be the behaviour of the second gene. The second gene could be high or as a result of a third gene could be still low. Therefore, the only conclusion about this type of relationship is that most of the time when the first gene is high, the second gene is low.

Dual interaction is the situation where sometimes a gene causes another gene to be high and sometimes low. We consider another type of common regulatory mechanism here, feedback loops which will be mentioned as a common motif in gene regulatory network in section 9.2. We consider a thermostat like operation. When one gene is high, another one is low and vice versa.

To summarize these regulatory relationships in terms of how we would expect to see the pattern in our data we provide Table 7-3 . The information provided in this table is based on RegulonDB and EcoCyc (Sun, Tuncay et al. 2007; Gama-Castro, Jimenez-Jacinto et al. 2008; Keseler, Bonavides-Martinez et al. 2009).

Table 7-3 Regulatory relationship and related patterns

Type of relationship	Pattern	Acronym	Percentage
Up regulation	High-High	ac	50%-60%of interaction
Down regulation	High-Low	re	30%-40% of interactions
Dual Interaction	High-Low & Low-High	du	5% of interactions

Having these patterns in mind, we designed a function to identify these patterns based on a two dimensional grid. We aimed to consider the expression values which frequently occurred in the upper or lower bounds of each gene pair in the same sample together. For this purpose we considered the first quartile and the third quartile of the values. A percentile is the value of a variable below which a certain percent of observations fall. The 25 percentile, known as the first quartile and is the value below which one quarter of the data is located. The 75 percentile, known as the third quartile, is the value that 75% of the numbers are below. In this way, we considered the upper and lower values that influence the relationships. In addition, by using such measures (quartiles) our function became less sensitive to the extreme values. The reason for that is for calculating quartiles we only rank values and choose top 25 percent or bottom 25 percent; therefore, we do not rely on the actual values and the shape of the distribution as such (Boslough and Watters 2008).

First we define a two dimensional grid. The two dimensional grid is a grid which has the discretized value of the first gene on the vertical axis and the second gene on the horizontal axis. The content of each cell inside the grid shows us the number of times that the first gene and the second gene in the same sample (record) have the expression values in the range of the cell boundaries. Figure 7-2 shows a typical two dimensional grid for G1 and G2.

G2 / G1	0.1-0.2	0.3-0.4	0.5-0.6	0.7-0.8	0.9-1
0.1-0.2	2	3	0	1	0
0.3-0.4	2	3	1	0	0
0.5-0.6	1	1	1	0	1
0.7-0.8	0	0	1	1	1
0.9-1	0	0	0	1	0

Figure 7-2 A two dimensional grid of G1 and G2

In Figure 7-2, the gray boundaries are discretized values usually between 0 and 1. However, the lower and upper boundaries are not necessarily 0 and 1 but depend on the maximum and minimum values of the gene. In the above examples, for simplicity, we provide an example where both genes have a similar range but in reality, for example, one gene might have a minimum of 0.3 and a maximum of 0.8 and the other might have a minimum of 0.2 and a maximum of 0.7. In such cases the boundaries for each gene starts with the minimum value of that gene. The cells inside the grid show us the number of samples that G1 is in that specific range and G2 is in another range. For example the first cell on the top left is 2 which mean two samples exist where G1 is between 0.1-0.2 and G2 is between 0.1-0.2 as well. The discretization assigns values to different bins and then we calculate the frequency of each bin which helps in visually depicting the relationship.

Now, with this two dimensional bin we can count the frequency of two genes occurring in the particular range. For example, we can count how many times when the first gene is more than the third quartile (the 75 percentile); the second gene is less than the first quartile (25 percentile). In other words, when the first one is high and the second one is low. We consider 75 percentile and 25 percentile as the cut boundary for being high and being low. We design the function in this way to be able to consider the definition of three major regulatory relationships and also for visualization of the relationship. Figure 7-3 represents an example of this two dimensional grid and the related calculation.

High Threshold=75 Quartile

Low Threshold=25 Quartile

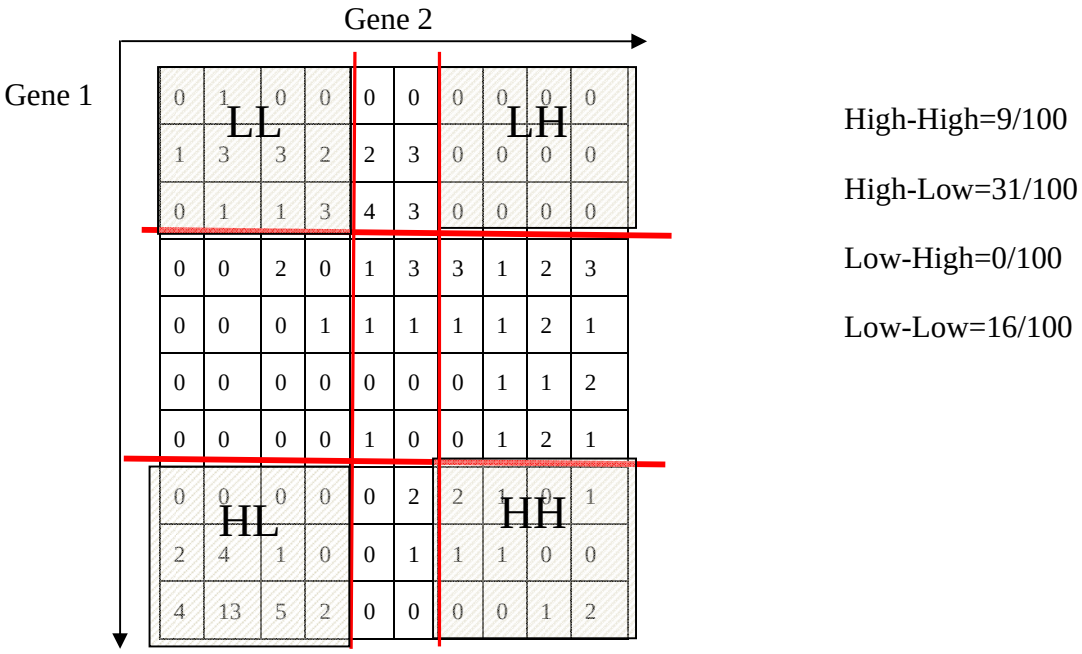


Figure 7-3 An example of 2D grid and calculation of the co-regulation function

The result of the analysis shows that there is a significant pattern of High-Low and therefore we can label this relationship as inhibitory or "re". The threshold to choose how many numbers are significant can be calculated based on the distribution of the data, but in general with testing many datasets having different gene distributions a threshold of 20 works well across the range of datasets.

Examples in GRN Literature:

The 2D Visualized Co-regulation function as proposed is a novel technique but there are some examples in the literature which have mapped a relationship between co-regulation and correlation in different ways (Allocco, Kohane et al. 2004). There is also a study which takes into account the state of transcription factor genes in the form of activated or inactivated, high or low inside a Bayesian Network approach (Noto and Craven 2005).

The reader should note that our proposed grid used in our co-regulation function is different from other grids used in this area, such as adjacency matrices. An adjacency matrix is a grid which has all the genes in each dimension and covariance correlation of each two genes as the content of each cell. The content of a cell is "1" if the distance is more than a threshold and "0" otherwise. In an adjacency matrix, the grid represents the relationship between all the genes, while in our case the grid just presents the relationship between two genes. In addition, the content of each cell is different in our grid, where each cell shows the frequency of having expression values of two genes on the same percentile range at the same time. Thus our grid has a different purpose and different contents and is proposed for the first time in this study.

Assumptions, Strengths and Limitations:

1. This function does not assume anything about the distribution of the data.
2. The function has a good visualization ability which gives experts the chance to see clearly the overall relationship between pairs and any existing patterns of relationship. The visualization makes the result plausible for the experts.
3. It has the ability to distinguish between different types of associations compatible with the definition of regulatory relationships.
4. Despite detecting the type of regulatory relationship still like Pearson or MI cannot detect the causality of the relationship. For example in up-regulation where we see a pattern of High-High, we can't say if gene A up-regulates gene B or the other way around.

7.6 Discussion

In the previous sections, we reviewed common association measures for gene regulatory network discovery, Pearson's correlation and Mutual Information. We reviewed the assumptions and limitations of each and arrived at the conclusion that there is no link between these similarity metrics and regulatory relationships such as up regulation and down regulation and dual interactions. This link needs to be defined in order to detect the

correct association. Here, we provide examples where other association measures fail to detect correct regulatory relationships.

Any relationship with the High-Low and Low-High patterns are detected by Pearson's correlation as a negative correlation and therefore, would be labelled as down regulation. However, Low-High is not a down regulation and only the High-Low pattern can be considered as a down regulation.

Mutual Information has an output value in the range of 0 to 1 and any number near to 1 is considered as an association and near to 0 as a non-association. This does not tell much about the type of the regulatory relationship. In other words, if the pattern is in the High-High, Low-High or High-Low the result is similar. Even if the pattern is in the middle like medium-medium still is considered the same. Therefore, MI does not tell us much about the type of the association. It indicates the existence of a relationship between a pair of genes, but does not specify the nature of that relationship.

We provided a list of desirable features for an association measure for GRN discovery and based on that we proposed an association measure to cover those features. We showed that our function is able to identify regulatory relationships that are impossible to detect otherwise. We also, showed the visualization ability of our function which provides a sense of the data for molecular biologists. Something important that we have to note is that the correlation or association between two variables does not necessarily mean causation. For obtaining such a relationship we need temporal dependency data (Boslough and Watters 2008).

In the next section, we will discuss a post-processing procedure which we designed to prune the result of our co-regulation function. Our post-processing procedure is a complementary process for our co-regulation function and improves its performance further. The post-processing procedure was also designed based on the heuristic information.

7.7 Post-Processing for False Positives

Usually the association measures detect some real answers and some false answers. Their performance not only depends on the high number of true detections but also on the small number of wrong detections. The correctly detected answers are known as true positives and the falsely detected ones are known as false positives. Table 7-4 explains the concepts of true positives, false positives, false negatives and true negatives. True positives are the number of pairs that we identified as the answer and they actually exist in the goal network. False positives are the records that we identified as the answer but they are not the answer, as they are not in the goal network. False negative means the number of records that we should detect, but we did not and true negatives are the pairs that are neither in the goal network nor in our experiments. True negatives indicate the number of pairs in the possible search space which are not our answer.

Table 7-4 Performance indicators of our experiments

		Goal	
		Found	**Not Found**
Result	**Found**	True Positive	False Positive
	NotFound	False Negative	True Negative

One of the biggest challenges for any association measure or any model aiming to discover dependent genes is the high number of false positives. The known regulatory associations can be used as positive training examples but obtaining negative examples is not straightforward, because definite knowledge that a given pair of genes do not interact is typically not available (Cerulo, Elkan et al. 2010). The number of false positive cases is very high using any method compared to true positives which result in a drop of performance considerably. Therefore, we tried to decrease the number of false positives by applying post-processing procedures. In this thesis, we applied two post-processing procedures.

The first one used heuristics information and the second one used a measure of information theory called Data Processing Inequality (DPI) (Cover and Thomas 1991). Based on the literature, indirect relationships happen where there is a chain between two genes and it usually result in a weaker relationship. The longer the chain is the weaker the interaction is. The direct relationship presents a strong gradual change (Akitaya, Seno et al. 2007).

DPI assumes that the false positives are the effect of indirect relationships; therefore are weaker connections compared to the direct ones. In a loop of connections between three genes DPI considers the weakest link as an indirect effect of other relationships and removes it if is considerably less than the other two. We applied DPI on the output of our co-regulation function to reduce the indirect relationships and improve the performance. More explanation about DPI will be provided in the next chapter where we will present the result of its related experiment.

We also proposed a heuristic post-processing. In our proposed co-regulation function, we looked only for the presence of a strong pattern of “ac” then “re” then “du” relationship which pass the threshold. Therefore, we did not consider information in other areas of the grid. Our proposed heuristic post-processing looks for the absence of a noticeable pattern of the opposite relationship in order to recognize a pair as true positive. Therefore any pair which presented a mild indication of the opposite relationship was considered as a false positive. It means when a pair is labelled as an “ac” then we look for the evidence of a “re” pattern and if it is labelled as “re” we look for the evidence of an “ac” pattern. Of course, the other pattern is not as strong as the pattern related to the current label and we need to set a different threshold for the reverse pattern. The reasoning behind this process is we know that strong gradual changes are indicator of a direct relationship (Akitaya, Seno et al. 2007); therefore, if we do not see a single strong pattern then it is more likely to be due to indirect influence. The heuristic post-processing was applied on the output of our co-regulation function in order to reduce the number of false positives and improve its performance further. Further explanation about our heuristic post-processing will be provided in the next chapter.

7.8 Summary

In this chapter, we reviewed common pairwise association measures for GRN discovery. We reviewed Pearson's correlation and Expected Mutual Information. We created a toy data set and presented the calculation of those measures on this data set. We also reviewed their limitations and assumptions. We concluded that existing association measures do not consider the definition of regulatory relationships to discover the regulatory relationships patterns. We also claimed that there is a lack of visualization ability in the existing functions. We presented a list of desirable characteristics for an association measure and proposed a new association measure that considers most of them. We designed our own association measure based on the definition of activation, inhibition and dual effect; the three major regulatory relationships. Activation is defined as: when one gene is high that causes the second gene to be high too. Inhibition is defined as: when one gene is high that causes the second gene to be low. For dual effects, we consider when a gene is high the second one is low and vice versa.

The new association measure is called 2D Visualized Co-regulation function and looks to find the regulatory relationships patterns in each gene pair. The co-regulation function has an innate ability to visualize the relationship as well. In the next chapter, we will provide experiments related to this function. We will also review the different versions of the function and will provide the result of the experiments for each version.

Chapter 8

Approach 3: Experiments with Co-regulation Function

"A scientific or technical study always consists of the following three steps:
1. One decides the objective.
2. One considers the method.
3. One evaluates the method in relation to the objective."

— Gen'ichi Taguchi

8.1 Introduction

In the previous chapter we reviewed some of the most common pairwise association measures for GRN discovery. We studied characteristics of each measure to find out what sort of pattern they are able to detect. We also raised the question of "What sort of relationship are we looking for?" in order to choose an appropriate measure which matches with the nature of the problem.

We found that none of the existing measures are able to exactly detect the three major regulatory relationships between genes. We provided a list of desirable characteristics which are needed to measure gene pairwise association. Later on, we designed our own association measure based on those characteristics. Our function was designed based on the definition of activation, inhibition and dual effect; the three major regulatory relationships. Activation is when one gene is highly expressed that causes another gene to be highly expressed. Inhibition is when one gene is highly expressed that causes another gene not to be expressed or to be expressed at a low level. The dual effect is when a gene sometimes activates and sometimes inhibits another gene.

Our association measure, called 2D Visualized Co-regulation function, passed three different stages of design. In the first step, we discretized values of each gene and categorized them into bins individually and then we considered the first quartile and the third quartile of each gene as the low and high boundaries in order to count the number of High-High and High-Low combinations. This version was based on one dimensional bin and was the simplest one.

In the second step, we calculated a two dimensional grid from two genes and then identified the regions in the grid which indicate up regulation, down regulation and dual regulation. In the third step, we tried to dynamically set the cut off threshold by identifying those regions dynamically. Moreover, we tried to show that our assumption about regulatory patterns in terms of machine learning was correct. We applied feature selection methods and also decision tree algorithm to our grid to find out the relationship between the class labels (ac, re, du) and boundaries of the densest area inside the grid.

In addition to our Visualized Co-regulation function and its related experiments, we did two post-processing experiments to reduce the number of false positives. The first post-processing procedure was based on heuristics and the second one was based on a measure from information theory called Data Processing Inequality (DPI).

All of these experiments will be reported in this chapter. Data for these experiments was produced by SynTReN, the synthetic data generator and was discussed in section 3.6. SynTReN provides us the facility to produce different networks. There are two types of sampling strategy in SynTReN: cluster addition and neighbourhood addition. The cluster addition method generates networks that are closer to the source network than the network generated with the neighbourhood addition. In addition to using these two different sampling methods we used different amounts of noise; biological noise and experimental noise; as well as different level of complex interactions. The first dataset was produced by the default parameters of SynTReN which employs a neighbourhood addition method to generate the network with some biological noise and experimental noise. The second dataset used the cluster addition method. The second dataset is the easiest because it was generated using the cluster addition method. The cluster addition

method generates a network which is more similar to the domain network and easier to discover compare to the neighbourhood addition. More details about the benchmark datasets can be found in the experimental setup in Section 3.6.

The algorithms in this chapter were implemented in Eclipse using Python as the programming language. We also used Python packages such as, NumPy and SiPy and StatsPy for matrix and numerical operations and statistical functions. RPy were used for accessing R and Bioconductor package. We also used *Minet* (Meyer, Lafitte et al. 2008) library from R to access the mutual information function implemented in ARCNE. Their function was reported to have the highest performance compared to other mutual information implementations (Meyer, Lafitte et al. 2008; Qiu, Gentlesa et al. 2009).

8.2 Experiment 1: Simple Frequency Based Co-regulation Function

Purpose: This was the first version of our co-regulation function. In this experiment, we aimed to see if defining a function based on the definition of regulatory relationship can perform as well as the other known measures in the area. The definition and implementation of the regulatory relationships was quite simple here. This version of the function was quite simple but was the starting point which later led us to the 2D version of the function.

Algorithm: This version of our co-regulation function works based on a simple idea. In the first step we considered all of the gene pairs and then we calculated the histogram (one dimensional bin) for each gene in a pair separately. Then we counted the number of High-High and High-Low and so on. The following steps will describe the procedure.

1. Calculate one dimensional bin (histogram) for each gene. Bins are equally divided.
2. Calculated 25^{th} percentile and 75^{th} percentile of each gene (any values above 75^{th} percentile are considered to be high and any values lower than 25^{th} percentile are

considered to be low). The choice of 25 percent comes from the experiment and is arbitrary.

3. Go through each sample and look for these relationships:
   3.1 High-High: Count the number of times (number of samples) that the first gene is high and the second one is high
   3.2 High-Low: Count the number of times (number of samples) that the first gene is high and the second one is low
   3.3 High-Low & Low-High: Count the number of times (number of samples) that the first gene is high and the second one is low, plus the number of times that the first gene is low and the second one is high.
4. If the total number of 3.1, above, is more than a pre-specified threshold, label the relationship as "ac" (up regulation)
5. If the total number of 3.2, above, is more than the pre-specified threshold, call this relationship "re" (down regulation)
6. If the total number of 3.3 is more than the pre-specified threshold called this relationship "du" (dual interaction)
7. Otherwise, label this relationship as "none" (no relationship)

The choice of threshold came from experiments. In our experiments we set the threshold to 17 percent. We tried this algorithm with one quarter and the third quarter of the mean instead of the median as well but the best result was achieved with median and percentiles. The justification for this is because the data is really skewed and does not have a normal distribution, median and related percentiles are the best descriptive statistics (Boslough and Watters 2008). Table 8-1 provides an example to describe the above procedure. In this table the expression values of the twenty samples were provided. The first quartile (25th percentile) and the third quartile (75th percentile) values were calculated and indicated by *Q1* and *Q3*. In each sample (record), if the value of the first gene was greater than *Q3* we labelled it as "High" and if was lower than *Q1* we labelled it as a "Low". The expression values between these were not important.

Table 8-1 A toy dataset and sample of calculations for simple frequency-based co-regulation function

	Ion	rpoH	Ion status	rpoH status
1	0.417949	0.142587		
2	0.843632	0.490734	High	High
3	0.560716	0.27895	High	High
4	0.41598	0.103687		
5	0.614831	0.2987	High	High
6	0.44987	0.2789	High	High
7	0.47191	0.156482	High	
8	0.35849	0.10288		Low
9	0.34258	0.25478		
10	0.425871	0.10498		
11	0.44987	0.27521	High	High
12	0.138224	0.112589	Low	
13	0.36987	0.008103		Low
14	0.317368	0.198754		
15	0.272448	0.105358		
16	0.23547	0.11022		
17	0.178962	0.047852	Low	Low
18	0.41897	0.00689		Low
19	0.32457	0.209607		
20	0.168776	0.047918	Low	Low
Q1	0.44987	0.259888		
Q3	0.306138	0.103485		

We then counted the number of times the first gene was “High” (greater than the third quartile) while the other gene was also “High” (greater than its third quartile). We called this number the High-High indicator. We did the same for the Low-Low, High-Low and Low-High. In this example the number of High-High was 5 and Low-Low was 2. As the total number of the record was only 20 and 5 out of 20 makes 25 percent of samples, therefore the pattern of High-High was dominant and based on these numbers; we recognized this relationship as “ac”.

Total number of High-High=5	Percentage of the High-High =5/20= 25%
Total number of Low-Low=2	Percentage of the Low-Low =2/20=10%
Total number of High-Low=0	Percentage of the High-Low =0
Total number of Low-High=0	Percentage of Low-High=0

We conclude that the first gene up regulates the second gene. The Pearson's correlation score for this pair was 0.741 while in most of the experiments the cut off threshold for Pearson's correlation is above 75%, therefore, based on Pearson's correlation these genes are not dependent. The situation with Mutual Information was different. The raw score was 1.1068 and after normalization, MI was able to just recognize the dependency of these two genes.

Setup: We used the first benchmark to compare the performance of this function against Pearson's correlation and Mutual Information.

Result: We applied our function on the first dataset and the result is presented in Table 8-2**.**

Table 8-2 Result of simple co-regulation function on the first benchmark

Method	**Total Records**	**False Positive**	**True Positive**	**False Negative**	**Recall**	**Precision**	**F-measure**
Mutual Information from *minet* Threshold=0.92	761	713	48	245	0.16	0.063	0.092
Pearson Threshold=0.9	516	422	42	250	0.14	0.081	0.10
Simple Frequency Based Co-regulation	710	672	38	255	0.13	0.05	0.08

In this experiment we described how our simple frequency based co-regulation function works and provided an example of its calculation. We also presented the result of applying this function to our first benchmark. This function was the first version (design) of our co-regulation function and the starting point which led us to a more sophisticated function. The second version of the function is based on a grid and comes with visualization ability. In the next section we will describe the second version of the function which we called Fixed 2D Visualized Co-regulation.

8.3 Experiment 2: Fixed 2D Visualized Co-regulation Function

Purpose: Here we describe the second version of our co-regulation function and provide the related experiments. In this version, we used a different way of discretization from that in the previous version. This way of discretization provided us with visualization ability. We called this function Fixed 2Dimensional Visualized Co-regulation function. In this experiment we present the visual impact of the function and also we test its functionality in terms of precision and recall against Pearson's correlation and Mutual Information.

Algorithm: The followings are steps of the algorithm:

Calculate a two dimensional grid of two genes as follows:

1. Discretize the first gene and the second gene in a fixed number of equal bins (for example, 10 bins).
2. Make a grid with the discretized values of the first gene on the vertical axis and the second one on the horizontal axis.
3. For each cell in the grid count the number of times that those genes have appeared together in one sample at those specific ranges.
4. Calculate the first and third quartile of each gene and then separate the following areas: High-High, High-Low, Low-High, Low-Low
5. Count the numbers of those areas referred to above.

   If High-High is more than a specific threshold, label this relationship as an "ac". Otherwise, if High-Low is more than a specific threshold, label this relationship as a "re".

   If neither of High-High and High-Low numbers were passed the threshold, consider the sum of High-Low and Low-High. If this number is greater than a specific threshold, label the relationship as a "du".
6. If the pair has not been labelled yet, label it as a "none".

Similar to the previous experiment, the choice of thresholds comes from the experiment. For this experiment we set the “ac” threshold as 18 percent and “du” as 25 percent. In the previous chapter we described this version of the function but in the following, we present another example to remind the reader about the process.

Table 8-3 A sample of expression values for rpoH and Crp and related quartiles

	rpoH	Crp
1	0.21021	0.838681
2	0.10452	0.874244
3	0.19246	0.589221
4	0.5473	0.115408
5	0.19212	0.471806
6	0.15109	0.53969
7	0.17842	0.801398
8	0.60287	0.011306
9	0.20815	0.649452
10	0.69288	0.006955
11	0.48004	0.121246
12	0.65649	0.091801
13	0.10563	0.782216
14	0.14524	0.753745
15	0.768	0.020388
16	0.06701	0.57206
17	0.74759	0.005891
18	0.32212	0.333972
19	0.06281	0.410009
20	0.10951	0.957171
…	…	…
Quartil	0.1502	0.217
Quartil	0.376	0.758
Max	0.961	0.995
Min	0.015	0.006

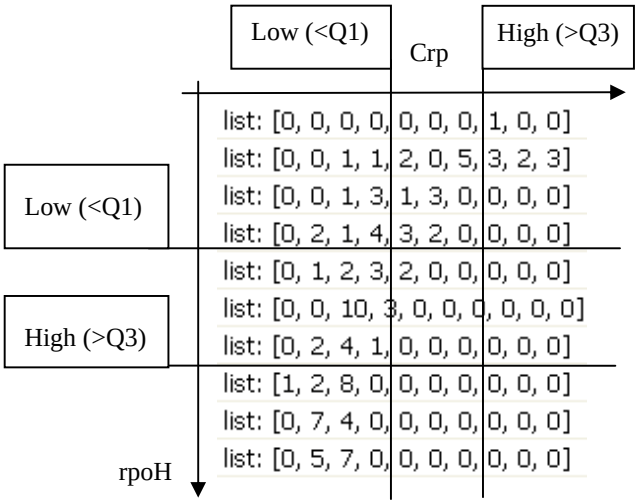


Figure 8-1 The two dimensional grid for crp and rpoH

Total number of records=100

Number of bins=10

HH=0

HL=34 & LH=9

LL=13

HL+ LH=43 ~ 43% > 25% ⇒ “du”

Figure 8-1 shows the two dimensional grid which was created based on these values. Each axis represents the discretized value of one gene. The content of each cell is the indicator of how many times those values occurred in one sample together. For example, in the upper most left corner cell we have zero, which means that we did not have any sample that the first gene was between its minimum value and the second bin's value, and also the second gene was between its minimum value and the next bin's value. After calculating this grid, we calculated the first quartile and the third quartile of each gene. The first quartile is the indicator of Low and the third quartile is the indicator of High. This means that any values less than the first quartile should be considered as Low and any values more than the third quartile should be considered as High. We drew these lines for each gene then we surrounded some areas which were indicator of two genes being Low-Low, High-High, Low-high and High-Low. In the last step, we counted the number of items in each of those areas and labelled this relationship based on those values. In the above example the label is "du".

After calculating this score for all gene pairs, we also applied a background correction process. In this procedure, we normalized the association values for each gene. Then we selected only those pairs displayed association values more than 0.5. This procedure helps to eliminate a number of false positives which are the result of having some genes which are correlated with too many other genes. A more sophisticated version of this background correction employed heuristic information as well. In that procedure, the degree of genes were extracted from the domain knowledge and then we normalized the association values for each gene, ranked them, and chose a number of top-ranked association values according to the degree of the gene. In the enhanced version of this heuristic background correction, we used a rule of thumb to improve the process. The rule was, if the degree of the gene was less than 15 we chose from the normalized ranked list exactly equal to the degree of the gene, otherwise we chose according to the 15+ 1/3(degree of gene-15).

Setup: We applied this function to several datasets to compare its performance with some of the well known association measures. We used datasets was produced by SynTReN which were discussed in Section 3.4

How does SynTReN work? We used the first, the second dataset and the sixth dataset here to observe the effect of different network topologies and also different amounts of noise. The first dataset used the neighbourhood addition method with some experimental and biological noise; the second dataset used the cluster addition method with the same amount of noise, and the sixth dataset uses the neighbourhood addition method without experimental and biological noise. The second dataset is easier than the first one, because of the cluster addition method and the sixth dataset is the easiest one because of not having experimental and biological noise. These datasets gave us the opportunities to see the effect of noise and the effect of the network topology on the result.

Result: For evaluating the performance of our method we used the F-measure which combines precision and recall together. Precision is a measure of exactness and is the number of the correct answers divided by the total number of answers that we found. Recall is the indicator of completeness and is defined by the number of correct answers divided by the number of the actual answers. F-measure is a measure to combine precision and recall and is defined by two times of precision and recall divided by the sum of them. More details about our performance measures were provided in Section 3.7.

Table 8-4 presents the result of this experiment on the first test bed, in terms of precision and recall and F-measure. The first dataset was generated using the neighbourhood addition method with experimental and biological noise of 0.1 and the probability of complex relationships set to 0.3. For each method and each dataset we applied different thresholds and reported the best performing threshold.

The result shows that our proposed function detected the same number of true positives compared with MI but far fewer false positives. In comparison with Pearson's correlation, our function yielded more true positives for comparable false positives.

Table 8-4 Result of experiments with the first benchmark

Method	Total Records	False Positive	True Positive	False Negative	Recall	Precision	F-measure
Mutual Information from *minet* Threshold=0.92	761	713	48	245	0.16	0.063	0.092
Pearson threshold=0.90	516	422	43	250	0.15	0.083	0.11
2D Fixed Co-regulation function Thresholds: du = 0.38; ac = 0.21; re=0.21	482	434	48	245	0.16	0.10	0.12

The second dataset was generated using the cluster addition method, probability of biological noise and experimental noise was set to 0.1 and the probability of complex 2-regulator interactions was set to 0.3. The second benchmark is easier than the first; therefore the performances of the functions are higher compared to that of the first benchmark as indicated in Table 8-5. Again in this dataset our function performance was superior. It produced the least number of false positives while it yielded fewer true positives than MI, but more true positives than Pearson.

Table 8-5 Result of experiments on the second benchmark

Method	Total Records	False Positive	True Positive	False Negative	Recall	Precision	F-measure
Mutual Information from *minet* Threshold=0.9	812	751	61	173	0.26	0.075	0.12
Pearson Threshold=0.85	598	546	52	182	0.22	0.087	0.13
Fixed 2D Visualized Co-regulation function Thresholds: ac=0.16 re=0.22 and du=0.32	570	513	57	177	0.24	0.1	0.14

Then we observed the effect of not having noise on the result by applying the function on the fourth and sixth data set.

Table 8-6 Result of experiments on the fourth benchmark

Method	Total Records	False Positive	True Positives	False Negatives	Recall	Precision	F-measure
MI threshold =0.95	776	704	72	221	0.25	0.093	0.14
Pearson threshold=0.9	543	477	66	227	0.23	0.12	0.16
Fixed 2D Visualized Co-regulation Thresholds: 0.18& 2 (for up and down) 0.38 for dual	502	430	72	221	0.25	0.14	0.18

The fourth dataset was produced without any biological noise but with experimental noise. The sixth data set was produced without any biological and experimental noise. The results for the fourth and sixth datasets are presented in Table 8-6 and Table 8-7. The sixth dataset is the easiest among the three as it does not contain biological and experimental noise; therefore the result shows a considerable improvement on performance compared with the other two datasets, especially the first dataset.

Table 8-7 Result of experiments on the sixth benchmark

Method	Total Records	False Positives	True Positives	False Negatives	Recall	Precision	F-measure
MI threshold =0.9	801	719	82	211	0.28	0.11	0.16
Pearson threshold=0.9	560	483	77	216	0.26	0.14	0.18
Fixed 2D Visualized Co-regulation Threshold= 0.18& 2 (for up and down) 0.38 for dual	553	437	87	206	0.3	0.16	0.21

The results of all the above experiments indicated that considering the definition of regulatory relationships in order to define an association function can improve the performance. This is mainly due to reduced false positive rates while maintaining a good discovery rate. Therefore, we can conclude that by using the Fixed 2D Visualized Co-regulation function we can achieve a higher success rate for genes pairwise associations.

Significance Test: We ran a paired t-test on the above results to find out if the results from co-regulation function on average are significantly better than those of MI and Pearson. We chose paired t-test as the changes in F-measure, precision and recall for the

different methods are not independent and change across the datasets accordingly. As paired t-test requires the assumption that there is no difference between the variances of two distributions we need to test our data for this assumption. We ran an F-test to see if the variances of the two distributions are the same and the result confirmed that there is not any significant difference between the variance of Pearson's distribution and that of the co-regulation function or that of MI and co-regulation function.

Table 8-8 Result of significance test for Experiment2

Compared Methods	Test	P-value	Is Significant
F-measure from Mutual Information against co-regulation	=TTEST({0.12,0.14,0.18,0.21},{0.092,0.12,0.14,0.16},1,1)	0.0069	Yes
F-measure from Pearson Correlation against Co-regulation function	=TTEST({0.12,0.14,0.18,0.21},{0.11,0.13,0.16,0.18},1,1)	0.018	Yes
Precisions from Mutual Information against Co-regulation function	=TTEST({0.1,0.1,0.4,0.16},{0.063,0.075,0.093,0.11},1,1)	0.0029	Yes
Precisions from Pearson Correlation against Co-regulation function	=TTEST({0.1,0.1,0.14,0.16},{0.083,0.087,0.12,0.14},1,1)	0.0009	Yes
Recalls from MI against Co-regulation function	=TTEST({0.16,0.24,0.25,0.3},{0.16,0.26,0.25,0.28},1,1)	0.5	No
Recalls from Pearson Correlation against Co-regulation function	=TTEST({0.16,0.24,0.25,0.3},{0.15,0.22,0.23,0.26},1,1)	0.019	Yes

The table indicates that in general our method achieved a significantly higher F-measure across different datasets compared with Pearson Correlation and MI. The improvements were mainly due to the increase in precision. The recall of our method was similar to that of MI and significantly better than Pearson correlation.

Case Study:

For the evaluation of the visualization effect of our co-regulation function, we carried out two case studies and asked two experts to evaluate our function. The experts that we chose both have many years of experience working with the gene expression microarray

data. The case study had three steps, in the first step; we described and illustrated how the function works by showing the experts a picture of the grid and explaining the algorithm steps. In the second step, we showed six training examples to the experts. The examples included, ac, re and dual relationships as well as no relationship. In the last stage, we tested the experts by presenting the expression profile of two genes at a time and asked the experts to guess what type of relationship it was. At this stage, we tested the experts by six examples and in the end, presented the result. Both experts said that it was not always easy to detect the type of the relationship by just looking at the expression values. They also both stated that the Co-regulation function was informative and helped them to better understand the nature of the relationship even without presenting them the numerical result of the function. The experts expressed their interest in seeing examples of applying the function to real datasets. The case study supported the useful visualization ability of the function.

8.4 Experiment 3: Feature Selection for Finding Important Areas of the Grid

Purpose: In this approach the aim was to test the validity of our assumptions about how the definition of the regulatory relationships would be reflected in the data. In doing this we tried to find the areas of the grid which are more important and most informative. For this purpose we applied feature selection algorithms to the grid. Feature selection algorithms can provide us with the list of the most relevant features for explaining the class labels to build a better predictive model of the data (Guyon and Elisseeff 2003).

Setup: For this experiment we used a variety of feature selection algorithms implemented in Weka package (Hall, Frank et al. 2009). For applying the feature selection algorithms we needed to transform the grid to a tabular format. This experiment was performed on a dataset that we made from the five initial datasets discussed in Section 3.6. From each of those five dataset some records were chosen randomly and total number of records was 358. We needed to perform this experiment on a combined

dataset, as we needed to extract general characteristics regardless of the type of the network. We computed the two dimensional grid for each pair and then we considered each cell in the grid as a feature in the dataset. Therefore, each row contained information related to cells in the grid and each column was an indicator of a single cell in the grid. Different rows represented different pairs. Each gene pair's grid transformed into one row. Figure 8-2 shows a grid and Table 8-9 provides an example of transformation of the data inside the grid to tabular formats. Each cell represents a feature (column) in dataset and the content of the cell is the value of that feature.

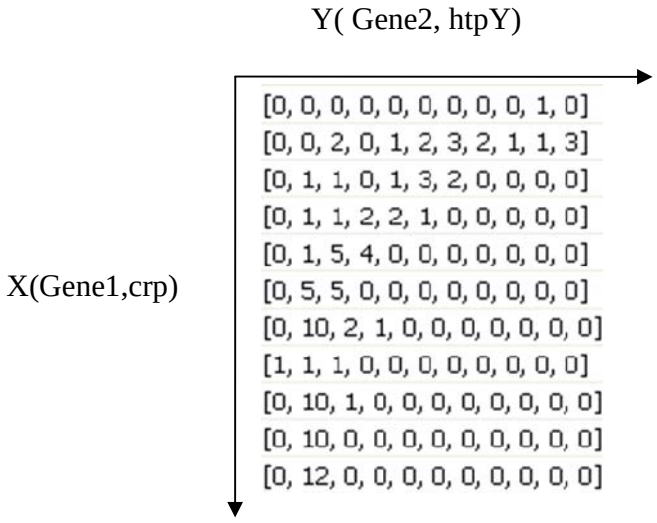


Figure 8-2 Example of two dimensional grid (bin Size=5) for crp and htpY

Then we added a final column to represent the class label for this pair. These labels are produced for our synthetic data by SynTReN. The aim was to use feature selection methods to find the most informative areas of the grid. In this way, we hoped to find more evidence to support our assumption about how the regulatory relationships are reflected in expression data.

Table 8-9 An example of data transformation from the grid for the feature weighting approach

x0 y0	x1 y0	x2 y0	x3 y0	x4 y0	x5 y0	x6 y0	x7 y0	x8 y0	….	x7 y9	x8 y9	x9 y9	x0 y10	x1 y10	x2 y10	x3 y10	x4 y10	x5 y10	x6 y10	x7 y10	x8 y10	x9 y10	x10 y10	Class
0	0	0	0	0	0	0	1	0	…	0	0	0	0	3	0	0	0	0	0	0	0	0	0	re

Algorithm: For the feature selection algorithm, we chose Gain Ratio Attribute Evaluator, Info Gain Attribute Evaluator and SVM Attribute Evaluator which are implemented in the Weka package (Hall, Frank et al. 2009). The results of these three methods were similar and had almost 85 percent overlap. Therefore, we only reported the result based on Gain Ratio Attribute Evaluator. The result was obtained by 10 fold cross validation re-sampling.

Result: The result is presented in the Table 8-10. The most important features are in the High-High area of the grid where we expect to see an "ac" pattern. Based on our experience, we expect to see that "ac" pairs show the clearest pattern. The second most important area is the High-Low area which belongs to "re" class labels.

Table 8-10 Result of the feature selection with Info Gain Attribute Evaluator

Rank	Attribute	Rank	Attribute
1	x10y1	19	x9y2
2	x10y10	20	x1y7
3	x9y10	21	x1y10
4	x9y1	22	x6y10
5	x9y9	23	x0y0
6	x10y9	24	x3y10
7	x8y1	25	x2y10
8	x8y9	26	x1y1
9	x9y8	27	x10y2
10	x8y2	28	x1y6
11	x1y9	29	x2y8
12	x2y9	30	x4y1
13	x4y10	31	x7y9
14	x7y1	32	x10y8
15	x5y10	33	x3y9
16	x7y2	34	x7y10
17	x10y0	35	x8y8
18	x1y8	36	x6y1

These features were mapped to the actual cells in the grid for a better representation. Figure 8-3 presents these important features inside the two dimensional grid. We found three distinct areas, when we mapped the features in Table 8-10 to the grid. These three areas were High-High, High-Low and Low-High. The High-High and High-Low areas confirmed our assumptions about pattern of "ac' (activation) and "re" (suppression)

relationships. The only issue in this experiment was the set of important cells (features) in the Low-High area which we could not explain, based on our assumptions. We assume that this pattern comes from how our synthetic data generator produces and considers "du" and "re".

This experiment almost demonstrated that our definition of regulatory relationships was valid. Other correlation functions, such as Pearson's correlation which relies on the orthogonal information only or Information theory measures, which consider any changes in any values, are not precise enough.

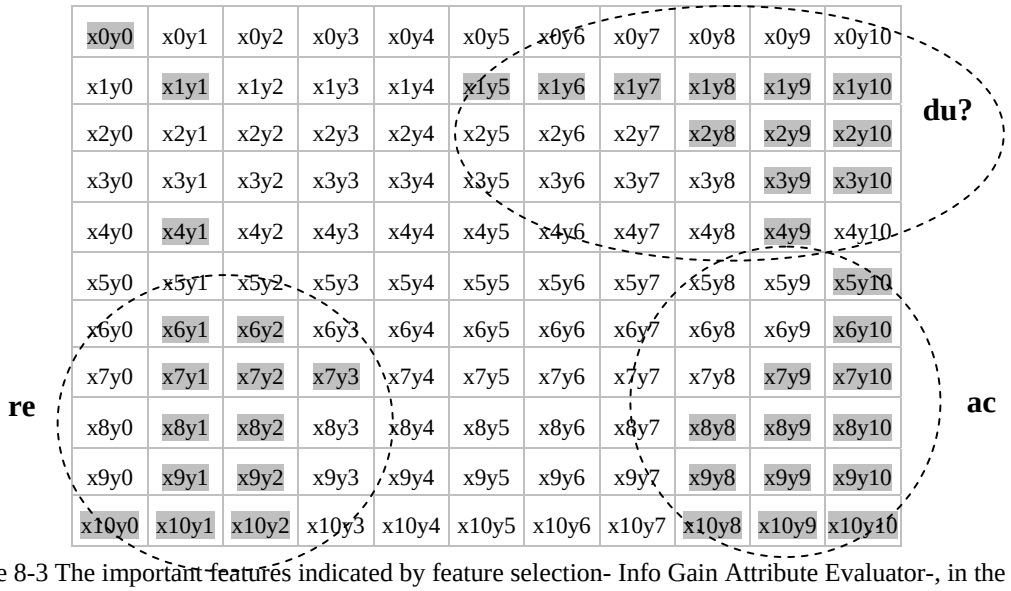


Figure 8-3 The important features indicated by feature selection- Info Gain Attribute Evaluator-, in the 2-dimensional grid representation

The observation of records with "du" class labels produced by our data generator showed us that this type of relationship has been defined in SynTReN like a flip-flop relationship, where one gene is low, another one is high and vice-versa. This pattern does not match the definition of "du" in the literature, where we expect to see both "ac" and "re" across different samples to recognize this as a "du" (Van den Bulcke 2009). The "re" patterns produced by SynTReN are also subject to further investigation. SynTReN produces a network with a similar structure to the real known networks, however, expression profiles are produced by SynTReN are not quite similar to known biological patterns (Li, Zhu et al. 2009). Just at the time of writing this thesis, a new GRN simulator was

introduced, called ReTRN, which was claimed to produce not only a similar structure, but also similar expression profiles (Li, Zhu et al. 2009). Our experiment should be repeated in the future on a real dataset or at least datasets generated by ReTRN for a further investigation of the exact regulatory patterns.

8.5 Experiment 4: Finding Rules with Decision Trees

Purpose: In this approach the aim was to test the validity of our assumptions about how the definition of the regulatory relationships would be reflected in the data. In other words, how we expect to see the pattern of regulatory relationships in terms of machine learning. In this experiment we approached this problem using a different technique. For this purpose, we decided to analyse the data with a decision tree algorithm to find general rules which may explain the relationship between the location and boundaries of the densest cluster inside the grid and the class labels (types of the relationship).

Algorithm: In doing this we modified our code for the Fixed 2D Visualized Co-regulation function and embedded a k-means clustering algorithm inside the code. For each gene pair, the algorithm first calculates the two dimensional grid as discussed earlier. Then it applies a k-means clustering algorithm to the grid in order to find the densest cluster inside the two dimensional grid and also the second densest if it exists (for dual relationship). The number of clusters (k) is set to three as we expect to see maximum of three clusters when the relationship is dual (one for each High-Low and Low-High and one for the rest of the numbers) and otherwise we expect to see two clusters (one is the densest cluster in either high-high or high-low and the second one is the rest of the grid). A good cluster has a small number of cells but a higher number of entries in the cells. Figure 8-4 shows an example of such a cluster. Then it calculates the density and the boundaries of the cluster in terms of minimum x, minimum y, maximum x and maximum y values. These x and y are the horizontal and vertical axes of the grid. Their range starts from 0 and ends with the maximum number of bins. For example, a cluster with a minimum x=5 and maximum x=9 and minimum y=1 and maximum y=5 is shown in Figure 8-4.

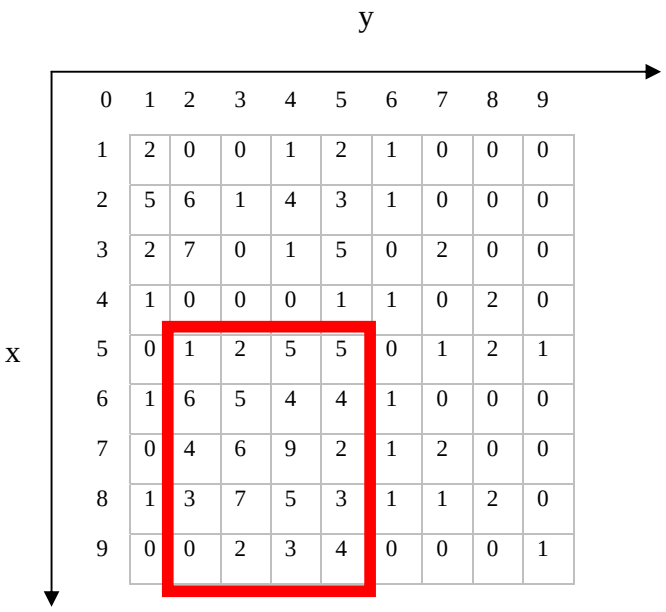


Figure 8-4 Example of the result of k-means clustering on a grid

The output of our program is in the tabular format as shown in Table 8-11. Each row contains a class label and associated minimum x, minimum y, maximum x, maximum y and density of the cluster inside the grid for each gene pair. Figure 8-5 describes the overall process.

Table 8-11 The output of applying k-means to the two dimensional grid

Class	MinX	MinY	MaxX	MaxY	Density
ac	0	7	9	10	1.265913
ac	9	0	10	5	1.382716
ac	0	5	10	7	1.449905
ac	3	0	10	6	2.9536
ac	0	0	5	7	2.964497
ac	0	8	7	9	3
ac	0	0	10	8	3.027778
du	0	0	4	4	2.270833
du	0	3	4	10	2.522222
re	6	0	10	4	1.398669
re	4	4	6	6	1.441082
re	0	0	7	8	2.913495
re	0	4	9	10	2.95679
re	3	0	9	7	3.160494

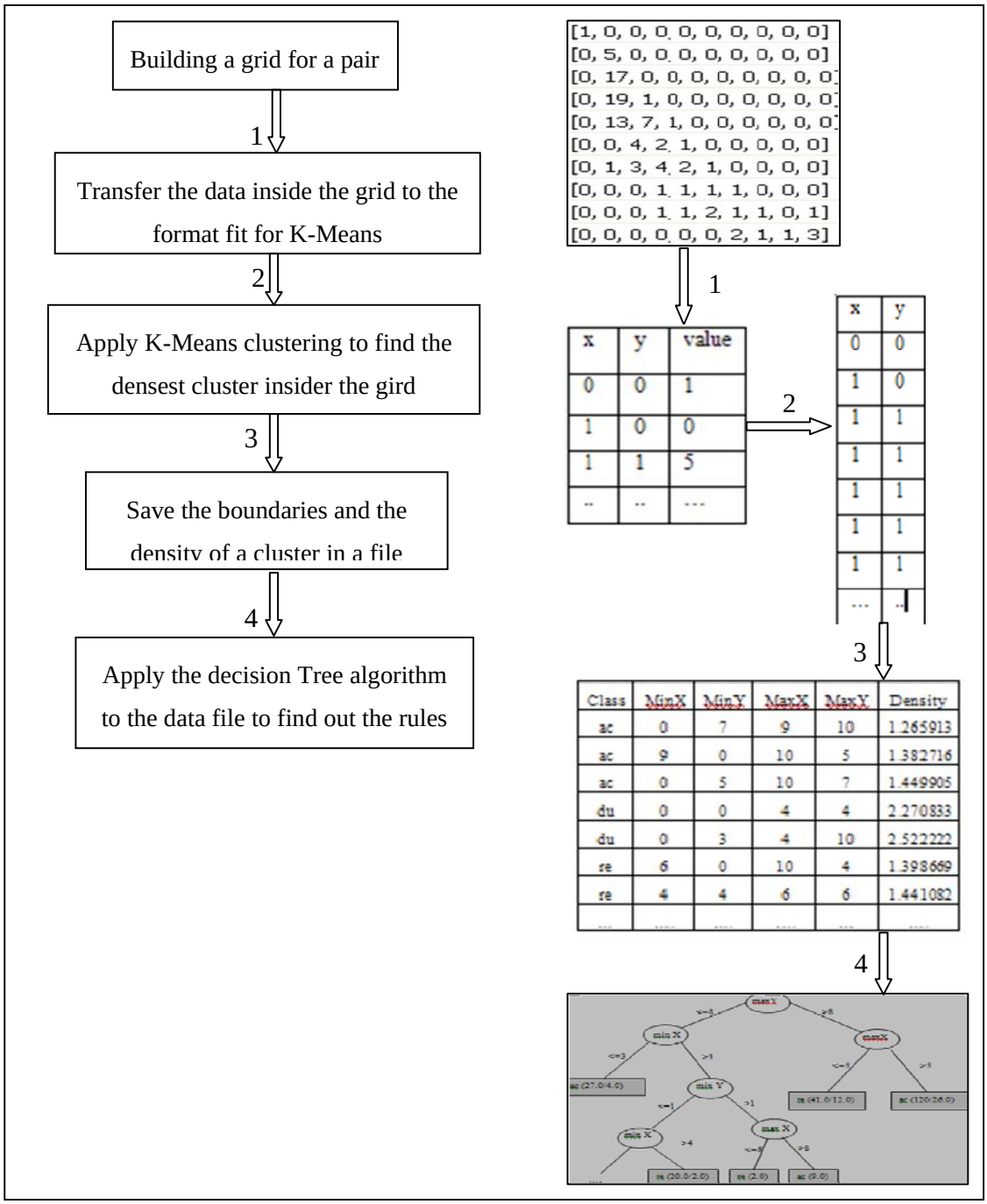


Figure 8-5 The process of finding rules with the decision tree

In summary, in this experiment we identified the minimum and the maximum of x and y of the restricted region and also the density of the region and used a data mining technique to identify any relationship between the above parameters and the class labels as a general rule. The data mining technique that was used here was decision trees. The aim was to discover general rules relating class labels to boundaries and density.

Setup: In this experiment we used the combined dataset which we used in the previous experiment. This dataset was made from the five initial datasets and total number of its record was 358. We needed to perform this experiment on a combined dataset, as we needed to extract general rules regardless of the type of the network. For the decision tree algorithm we used the J48 and Random Forest algorithms in the Weka package (Hall, Frank et al. 2009). For the k-means algorithm we used an implementation of k-means in R (Fraley, Raftery et al. 2006).

Result: The boundaries of clusters plus their density and the class labels were given as the input to a decision tree algorithm to find out the rules describing the class labels, based on the boundaries of the clusters. The output rules are presented in Figure 8-6.

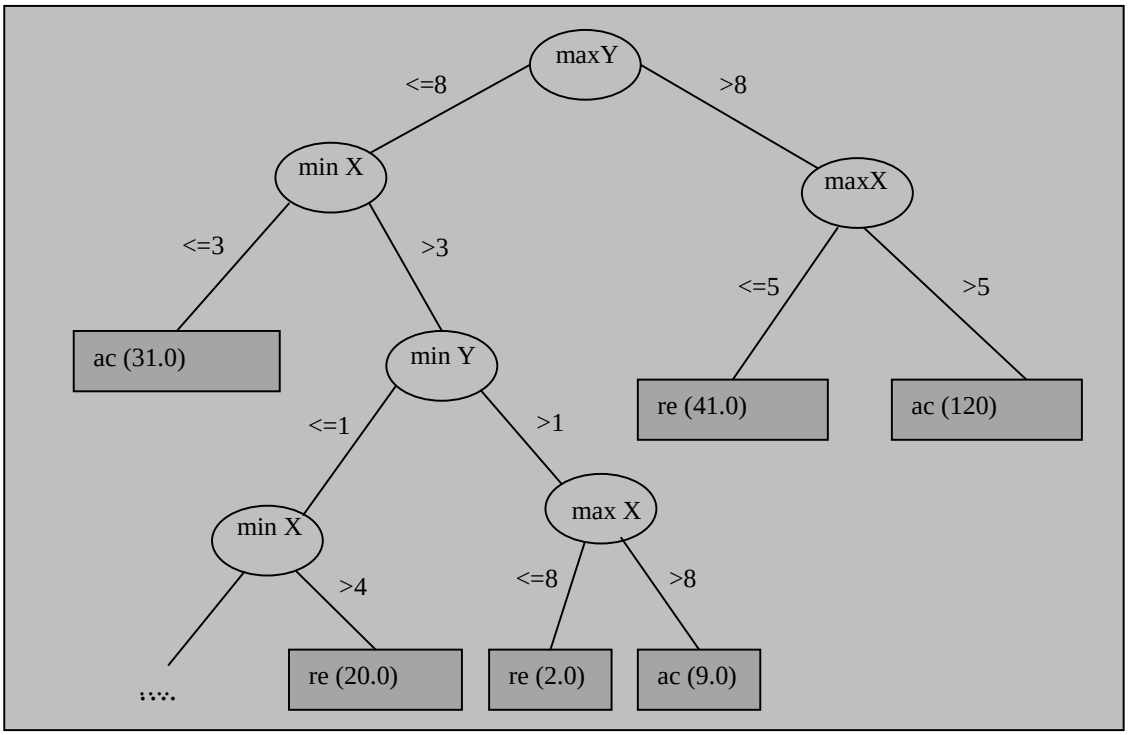


Figure 8-6 Decision tree J48 for rule discovery based on the boundaries of clusters

For simplicity, in Figure 8-6 we only kept the important branches. The tree shows the rules related to “ac” and “re” labels. There are two main rules for “re” and one for “ac”. As we mentioned earlier, in the previous experiments the “re” labels produced by our synthetic generator did not completely match with the definitions in the literature, therefore we assume that the second set of rules for “re” happens because of this property of synthetic data generator. The important cells for detection of “re” and “ac” classes are highlighted inside the two dimensional grid in Figure 8-7

x0y0	x0y1	x0y2	x0y3	x0y4	x0y5	x0y6	x0y7	x0y8	x0y9	x0y10
x1y0	x1y1	x1y2	x1y3	x1y4	x1y5	x1y6	x1y7	x1y8	x1y9	x1y10
x2y0	x2y1	x2y2	x2y3	x2y4	x2y5	x2y6	x2y7	x2y8	x2y9	x2y10
x3y0	x3y1	x3y2	x3y3	x3y4	x3y5	x3y6	x3y7	x3y8	x3y9	x3y10
x4y0	x4y1	x4y2	x4y3	x4y4	x4y5	x4y6	x4y7	x4y8	x4y9	x4y10
x5y0	x5y1	x5y2	x5y3	x5y4	x5y5	x5y6	x5y7	x5y8	x5y9	x5y10
x6y0	x6y1	x6y2	x6y3	x6y4	x6y5	x6y6	x6y7	x6y8	x6y9	x6y10
x7y0	x7y1	x7y2	x7y3	x7y4	x7y5	x7y6	x7y7	x7y8	x7y9	x7y10
x8y0	x8y1	x8y2	x8y3	x8y4	x8y5	x8y6	x8y7	x8y8	x8y9	x8y10
x9y0	x9y1	x9y2	x9y3	x9y4	x9y5	x9y6	x9y7	x9y8	x9y9	x9y10
x10y0	x10y1	x10y2	x10y3	x10y4	x10y5	x10y6	x10y7	x10y8	x10y9	x10y10

Figure 8-7 Representation of the output of the decision tree on the 2-dimensional grid (light grey is re, dark grey is ac)

The light grey represents the area related to “re” and the dark grey ones represent related area to “ac”. This experiment confirmed the result of the feature selection experiment. Thus these experiments validated our assumption about regulatory relationships in terms of machine learning.

8.6 Experiment 5: Variable 2D Visualized Co-Regulation Function Using a Sliding Window

Purpose: We analysed pairs that we were incorrectly detected by the Fixed 2D Visualized Co-regulation function. We discovered in many cases there was a pattern, but it was shifted out of the restricted area and that is why we could not detect those pairs. Therefore, it might be beneficial, to not have a fixed restricted area. If the area of interest can be dynamically restricted, then a fixed threshold is not required anymore and our method will be more flexible and general.

In doing this, we tried different strategies. The first strategy used a sliding window over the grid to find an area with the maximum frequency numbers and the minimum number of cells. Considering an “ac” relationship the search area for identifying “ac” started from the bottom right corner (maximum, maximum point) until the Low-Low point (a

conjunction of first quartile lines for the first and second gene). For "re" relationship the area starts from the bottom left corner (maximum-minimum) point to the Low-High boundary which is the conjunction point of the first quartile of the first gene and the third quartile of the second gene. For clarity, the search directions are presented in Figure 8-8.

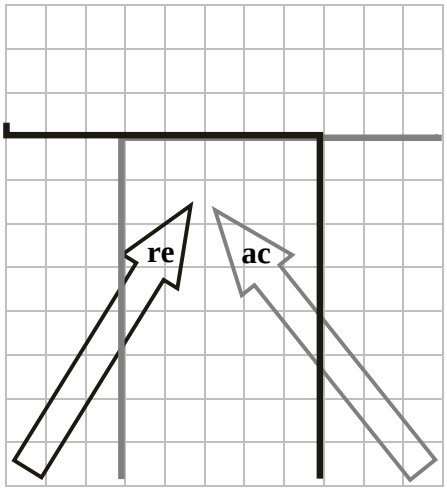


Figure 8-8 Search directions for the sliding windows over the grid for "ac" and "re"

Algorithms: This algorithm starts from the predefined fixed area. It calculates the score of each pair according to the numbers in High-High, Low-Low, High-Low and Low-High areas. This step is similar to what we did in the fixed version. If a pair does not earn enough score to be labelled then we will go to the second step. In the second step, we expand x and y boundaries, one at the time and then we calculate a score which indicates the density of the area based on the following formula:

$$\frac{Sum\ of\ the\ Area}{Total\ Number\ of\ Records} - 2 * \frac{Number\ of\ Cells\ in\ the\ Area}{BinSize^2} \tag{8.1}$$

If this metric for the new area is greater than the previous area we keep the new boundaries, otherwise we try to expand the boundary again by 1. In this algorithm the number of tries for expanding the boundaries is a flexible parameter. We tried different

values for this parameter and the best result was achieved with four tries. We also tried different formulas to find the best formula to describe a dense cluster.

Setup: We tried this algorithm on the first dataset to see if it was going to make any difference. The best result was obtained with the four tries around the area, which is presented in the Table 8-12.

Table 8-12 Result of sliding threshold co-regulation function

Method	Total Records	False Positives	True Positives	False Negatives	Recall	Precision	F-measure
Fixed 2D Visualized Co-regulation	482	434	48	245	0.16	0.10	0.12
Sliding 2D Visualized Co-regulation	719	570	56	237	0.19	0.078	0.11

Result: This strategy did not result in improvement because we could not find the best way or formula to effectively calculate that area. Therefore, we decided to follow another approach. In the next section we will describe another technique for automatically isolating the area of interest using machine learning methods. We called it black box modelling. In the black box modelling we tried to solve the problem by using a data mining technique to detect the pattern of regulatory relationships automatically. We performed this experiment on a combination of our five datasets plus some artificial nodes. This database was created to have the highest variability, as this way the machine learning algorithm could be trained to distinguish the pattern of regulatory relationships better.

8.7 Experiment 6: Dynamic 2D Visualized Co-regulation Function Using Black Box Modelling

Purpose: In this experiment the aim was to build a model to learn the pattern of regulatory relationships from an ensemble of datasets. Such a model will remove the need for finding any cut off threshold. This way the trained model would detect the

relationships automatically. The result of our previous experiments showed us that if we set up a way to find the thresholds dynamically we could improve our function further.

Setup: For this experiment, we organized a training dataset and a test dataset. The training set contained records from each of the initial five datasets. We also added some "none" pairs. In addition to these records we added some artificial records that we created based on our definition from the regulatory relationship. These artificial records were designed to show the exact pattern of the regulatory relationships. We created these artificial records in order to test whether learning these patterns might result in better performance compared to when such a pattern is not learnt.

The training dataset that we used for this experiment finally contained 518 records. The test dataset was the 6th microarray test bed, which had the same genes but different characteristics and a different network. The test dataset contained all possible combinations of the 200 genes, which is 40,000 records and covers the possible search space. Table 8-13 shows an example of the dataset used in this experiment.

Table 8-13 Examples of records in training and test datasets

G1	G2	x0 y0	x1 y0	x2 y0	x3 y0	x4 y0	x5 y0	x6 y0	…	x8 y0	x9 y0	x10 y0	x10 y5	x6 y10	x7 y10	x8 y10	x9 y10	x10 y10	Class
rpoH	lon	1	0	0	0	0	0	0	…	0	0	0	0	0	0	0	1	1	ac
rpoH	mopA	0	1	0	0	0	0	0	…	0	0	0	0	0	0	0	0	1	ac
rpoH	grpE	1	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0	1	ac
rpoE_rseA	rpoH	0	0	0	1	0	0	0	…	0	0	0	1	0	0	0	0	1	Ac
rpoE_rseA	ecfG	1	0	0	0	0	0	0	…	0	0	0	0	0	4	4	7	13	Ac
rpoE_rseA	rpoD	0	1	0	0	0	0	0	…	0	0	0	0	0	0	3	7	12	Ac
crp	rpoH	0	0	0	0	1	0	0	…	0	0	0	2	0	0	0	1	0	Du
crp	crp	1	0	0	0	0	0	0	…	0	0	0	0	0	0	0	0	12	Du

Algorithms: We applied different classification methods such as Naïve Bayes and Random Forest to our datasets to see how they learn the model. Then we saved the output model and presented the test dataset to this model. We measured the performance of the model on the training and test dataset. Among all the classifiers, we achieved the

best performance with KStar (Cleary and Trigg 1995). We used the implementation of the above algorithms in Weka (Hall, Frank et al. 2009).

Result: We had two sets of results. The first one belonged to the initial training dataset and the second one belonged to the second training dataset. The second dataset was created from the first dataset by adding more none records. For each training dataset we reported the performance on the test set as well.

The initial training dataset had 435 records. Out of this, 358 were randomly selected from the five target networks and the rest were artificial records that we made to teach the exact pattern of "ac" and "re" and "du" to the classifier. Table 8-14 presents the result of such a classification on training and test set. The result is presented in the form of a confusion matrix. The confusion matrix is a matrix used to summarize the results of a classification. Items along the main diagonal are correct classifications and other than those ones are errors (Bramer 2007).

Table 8-14 Confusion matrices of the first training set (left) and the first test set (right)

ac	du	re	none	Classified as
215	7	18	10	ac
7	7	6	2	du
31	7	65	9	re
11	2	4	76	None

ac	du	re	none	Classified as
162	3	26	9	ac
2	6	2	2	du
25	5	50	1	re
19067	600	10794	9240	none

The performance on the training set was 76% and on the test set was 23.6%. Then, we added 20 more artificial "none" records to the training set in order to teach the model the rejection pattern ("none") better. The effect of this was that the amount of the false positives was decreased therefore the performance increased. Then we added another 20 none records and the same affect was recorded and the result improved further.

Table 8-15 Confusion matrices of the final training set (left) and the final test set (right)

ac	du	re	None	Classified as
206	6	20	18	ac
6	7	6	3	du
27	7	62	16	re
13	1	9	110	none

ac	du	re	None	Classified as
149	3	26	22	ac
2	6	2	2	du
23	5	47	6	re
16701	541	10063	12396	none

The result after adding "none" records on the training dataset: Table 8-15 shows the performance of the model on the final dataset. We achieved 91 percent accuracy on the training data however; again in the test dataset performance was not that good. The performance was about 32 percent. The drop in performance happens mostly because of having different number of records in training and test datasets, especially "none" records. This result shows performance improvement by adding more artificial "none" records to the dataset. By adding more "none" records we helped the classifier to learn a variety of none relationship patterns more easily. Since "none" relationships are the most frequent relationships this can improve the performance of the classifier considerably. It can be observed that the size of training and test datasets here are still very different and by adding to the size of the training set we can train the classifier to perform better.

The result on the test dataset: As it shows in Figure 8-9, the performance was dropped to 31.50 percent. As the confusion matrix shows in Table 8-15, there is still a good accuracy on the "ac", "re" and "du" labels but the drop in performance is due to "none" labels.

```
C:\Program Files\Weka-3-4> java -classpath weka.jar weka.classifiers.lazy.KStar
-l C:\Armita\MyProgram\Experiments\WithWeka\1+2+3\Model1.model -T C:\Armita\MyPr
ogram\Experiments\WithWeka\1+2+3\BinOutput5.arff

KStar Beta Verion (0.1b).
Copyright (c) 1995-97 by Len Trigg (trigg@cs.waikato.ac.nz).
Java port to Weka by Abdelaziz Mahoui (am14@cs.waikato.ac.nz).

KStar options : -B 20 -M a



Correctly Classified Instances       12598               31.4997 %
Incorrectly Classified Instances     27396               68.5003 %
Kappa statistic                          0.0057
Mean absolute error                      0.3422
Root mean squared error                  0.5671
Total Number of Instances            39994

=== Confusion Matrix ===

     a     b     c     d   <-- classified as
   149     3    26    22 |     a = ac
     2     6     2     2 |     b = du
    23     5    47     6 |     c = re
 16701   541 10063 12396 |     d = none
```

Figure 8-9 The result of the black box modelling experiment on the test dataset

The test set contained all possible gene pairs from the sixth dataset (size 40,000 pairs). The training dataset size contained 600 records that were collected from the target networks of our five initial datasets. In addition to those records we added some artificial records which helped the program to learn the model of “ac”, “re” and “du”. We compared the performance of the model when we did not present these artificial records with the performance when we did present these records. We achieved a better result with presenting these artificial records, which is an indication that learning those regulatory relationships can help to improve the procedure. This once again confirmed our previous result which proved to us that our interpretation of regulatory relationships is valid.

This experiment indicated that we could use machine learning to learn the pattern of regulatory relationships and set the threshold dynamically. In this experiment we did not achieve a high performance, as we had a high number of false positives. For increasing the performance we needed to add more “none” records to the training dataset, as the drop in performance was due to high number of false positives. Our data was limited to

the known network of E. coli; therefore we could not add more extra known records to increase the performance.

Another way for further improvement could be to change the encoding in such a way that close cells which are related to each other become close features in the dataset as well. In this way, the classifier can learn the pattern of relationships better. Figure 8-10 describes the effect of encoding on data transformation. The left grid has a distinct pattern on its top left corner, but when we read the cells in sequential rows and transform it to a record the pattern has vanished and there is not much difference from a classifier perspective between the left grid and the right grid.

1	2	4	0	0
2	1	0	0	0
3	0	0	0	0
…	…	…	…	….

A grid with a pattern on the top left side

1	2	4	0	0
0	0	0	0	0
0	0	0	0	0
…	….	….	….	….

A grid without that pattern

	F00	F01	F02	F03	F04	F10	F11	F12	F13	F14	F20	F21	F22	F23	F24
Left Grid	1	2	4	0	0	2	1	0	0	0	3	0	0	0	0
Right Grid	1	2	4	0	0	0	0	0	0	0	0	0	0	0	0

Figure 8-10 Example of two grids and the effect of transformation on the pattern

In addition to encoding, we could make further improvement by building the grid in a more effective way. One way would be to use a better discretization method which is finer where there is more information and that could help by increasing the amount of given information and subsequently increases the performance of the classifier.

In the next experiment, we will describe how we used heuristics to create a post-processing procedure to reduce the number of false positives and increase the performance of GRN discovery.

8.8 Experiment 7: Heuristic Based Post-Processing

Purpose: The high number of false positives is the main challenge for any GRN discovery algorithm and is a well known problem in the literature. This phenomenon happens because we cannot distinguish indirect relationships from direct relationships. The indirect effect happens because genes influence each other through other genes and sometimes this indirect effect represents a stronger correlation than the direct effect.

In the previous chapter we described how we intended to use heuristic information for eliminating the false positives. Here we set up an experiment to test this idea. This experiment tests whether using information regarding the absence of the other types of relationships can reduce the number of false positives and improve the performance.

The elimination process is done by looking for the absence of the reverse relationship. In this experiment, we only applied this procedure to the self loops. Self loop is a mechanism where a gene is acting on itself. The self loop genes usually are transcription factors the product of which binds to its own promoter. Self-regulation is a key-lock relationship and usually represents a reversible reaction like when an enzyme consumes its own product (Goemann B, Potapov AP et al. 2009). Self-regulation is particularly important and is our main focus here as the other methods are usually not able to detect self-loops. Self-regulation makes as many as 59% of the transcription factors in Escherichia coli. This means 59% of genes regulate their own transcription rate (Hermsen, Ursem et al. 2010). Out of the total number of self-regulation, 87% are negative feedback, 6.5% are positive feedbacks, and 6.5% are dual circuits(Thieffry, Huerta et al. 1998).

Another reason to apply this procedure only to self loops was that after the initial experiments we decided to limit our experiments to a specific condition. This enabled us to measure the effect of that condition directly. Therefore we limited the experiment to only self loops. Further experiments could be done to measure the effect of this procedure in general on any pairs.

We analysed 17 cases to discover a rule and then we implemented this rule in a program to see the result on the whole dataset. The rule which we discovered for self-regulation is based on the contents of cells in the High-High and Low-Low area. The example in Figure 8-11 helps the reader to understand the post-processing step. The gene in this example is “fur”.

1	0	0	0	0	0	0	0	0	0
0	2	0	0	0	0	0	0	0	0
0	0	2	0	0	0	0	0	0	0
0	0	0	5	0	0	0	0	0	0
0	0	0	0	6	0	0	0	0	0
0	0	0	0	0	16	0	0	0	0
0	0	0	0	0	0	18	0	0	0
0	0	0	0	0	0	0	23	0	0
0	0	0	0	0	0	0	0	17	0
0	0	0	0	0	0	0	0	0	10

Figure 8-11 A negative example for illustration of post-processing on self-regulatory relationships

Here we see a clear example of High-High as the numbers in the Low-Low area are much weaker than High-High. Therefore, we recognize this as a valid pair. In contrast another example is “dnaA”. As we can see in Figure 8-12 the numbers are almost evenly distributed across the diagonal. This pair was recognized as a self-regulation by our Fixed 2D Visualized Co-regulation function because of having high numbers in High-High area. However, because the numbers in the Low-Low area are considerable, we label this as a false positive.

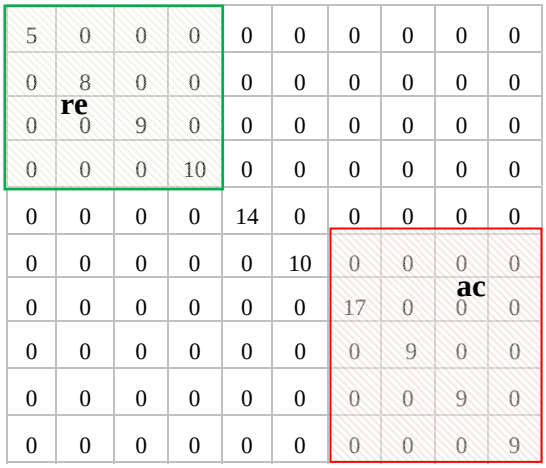


Figure 8-12 A positive example for illustration of post-processing on self-regulatory relationships

Table 8-16 shows the 15 gene pairs that we analysed in terms of the contents of High-High and Low-Low area. The pairs on the left are true positives. The pairs on the right side are pairs that were previously recognized as "ac" self-regulations but they are not, as they have a considerable pattern of Low-Low as well.

Table 8-16 The result of analysis of self-regulatory pairs. The left table shows the true positive self-regulation and the right table shows the false positives.

High-High	Low-Low	Ratio(LL/HH)
39	12	0.31
50	5	0.1
37	8	0.22
41	21	0.51
43	13	0.3
42	15	0.36
44	12	0.27

High-High	Low-Low	Ratio(LL/HH)
29	8	0.28
37	24	0.65
34	25	0.74
23	22	0.96
83	6	0.06
2	55	27.5
5	62	12.4
21	38	1.81

The rule which we extracted for detecting a false positive based on the above tables is as follows:

$$0.1 < \frac{Number\ of\ Low\_Low}{Number\ of\ High\_High} < 0.6 \qquad (8.2)$$

This simple rule was extracted from 15 cases in Table 8-16 (7 true positives and 8 false negatives) and it can distinguish all of the self regulatory false positive pairs. This rule simply says that when the numbers in High-High area are considerably higher than in the Low-Low there is more chance of having a true self-regulatory up regulation.

The result of this case study is consistent with our assumption about the regulatory relationships described in Chapter 7.

Setup: We used the output of the experiment mentioned in Section 8.7 as the input for this process. The input for this procedure is the result of our Fixed 2D Visualized Co-regulation function mentioned in the previous sections on the fourth benchmark.

Algorithm:

- Read the final pairs result of the Fixed 2D Visualized Co-regulation function.
- Calculate a two dimensional grid for each pair in that list which is a self loop.
- Count the number of items in the High-High area and also Low-Low area. Calculate the proportion of Low-Low to High-High according to the Equation (8.2).
- If the result is more than a threshold, consider the pair as a false positive and remove it from the final list.

Result:

Table 8-17 The result of heuristic post-processing

	True Positives	False Negatives	False Positives	Recall/Precision	F-measure
Before	72	221	430	0.25/0.14	0.18
After	72	221	416	0.25/0.15	0.19

Here the threshold for the proportion of the Low-Low's to High-High's was set by experiments. This procedure was so precise, with 100% percent accuracy. It detected 12 pairs and all of them were false positives.

Significance Test: We performed a binomial test to find out if the results after and before applying the heuristic post processing are significantly different. The binomial test here, considers each detection as a true or false discovery. The calculation was done using Matlab 7.11 implementation. Equation 8.1 shows the formula for calculating the binomial tests.

$$f(x,n,p) = \binom{n}{x} p^x q^{n-x} I_{(0,\ldots,n)(x)} \tag{8.3}$$

$$Before: y = \binom{641601}{72}\left(\frac{293}{641408}\right)^{72}\left(1-\frac{293}{641601}\right)^{641601-72}$$
$$= binopdf(X,N,P) = binopdf\left(72,723,\frac{293}{723}\right) = 2.12\, e^{-75}$$

$$After: y = \binom{641601}{72}\left(\frac{293}{641601}\right)^{72}\left(1-\frac{293}{641601}\right)^{641601-72}$$
$$= binopdf\left(72,709,\frac{293}{709}\right) = 5.095\, e^{-76}$$

We can observe significant improvement in the result after applying the post-processing step therefore we concluded that our post-processing step was a very effective process. The result of this experiment once again confirmed the effectiveness of using heuristics. It also validated our assumption about the pattern of regulatory relationships. Further improvement would be achieved by investigation of applying the same principle for the post-processing of the other pairs, not only the self-regulators.

8.8.1 Experiment 8: Proof for Heuristic Post-Processing Using Weights of a Neural Network

Purpose: The aim of this experiment was to confirm the theory behind our heuristic post-processing by finding evidence from the expression data. Our assumption was that the presence of the opposite relationship is informative and can help to further

distinguish classes. For example, a pair which has been recognized as an “ac” is a true positive if it does not have considerable content in the “re” area.

To confirm this assumption, we employed a neural network to find cells inside the grid which had negative weights in relation to each class label. Cells with negative weights in relation to a class label are informative, like cells with positive weights. Having considerable entries in cells with positive weights in relation to a class label can confirm the presence of that class. Similarly, having entries in cells with negative weights is an indicator of having weak dependency on that class label. In this experiment, we first found cells with negative weights for each class label. Then we looked to see in which areas of the grid they were located. We found they were located in those areas related to the opposite relationship. This meant that the assumption behind our heuristic post-processing was valid.

For this purpose we used a simple neural network called multilayer perceptron with no hidden layer to discover the internal weights. We did not use a hidden layer of neurons, as this enabled us to find direct relationships between inputs and outputs. Neural networks express this relationship by assigning a weight (either positive or negative) to each connection from an input node to an output node.

The reason behind choosing a neural network for this experiment was that there are studies which confirm that inter node weights of neural networks can indicate the importance of features (Sestito and Dillon 1991; Sestito and Dillon 1993). In addition, neural networks can provide us with negative weights as well as positive weights for each feature in relation to each class label. This gives us an opportunity to recognize the presence of high value entries in those cells with negative weights as an indicator of false positives.

Figure 8-13 represents the idea of using a neural network for the discovery of important features for each class label. We had three classes which made three output nodes for the neural network and we have some input nodes which were the cells inside the grid. The features’ weight in each node shows how important those cells are for that class label.

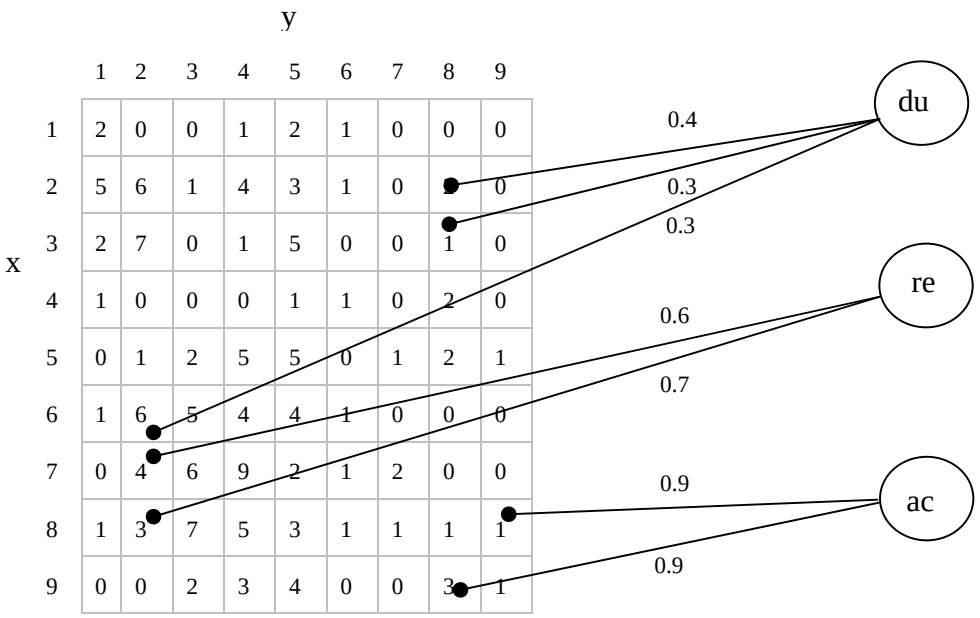


Figure 8-13 Neural network for discovery of the important cells inside the grid

Setup: This experiment was performed on a dataset that we made from the five initial datasets mentioned in 8.2. We computed a two dimensional grid for each pair and then we applied a transformation process similar to what we did in the feature selection experiment. The data transformation process enabled us to apply a neural network to the data. The data after transformation was in a tabular format, where each column was a single cell and each row had all the cells related to a pair. Different rows represented different pairs. In this way, each cell inside the grid became a feature in the dataset, and the neural network with no hidden layer was applied to find the direct relationships between cells (input nodes) and the class labels (output nodes). Figure 8-14 shows a small grid and Table 8-18 provides an example of such a transformation of the data from the grid to a tabular format.

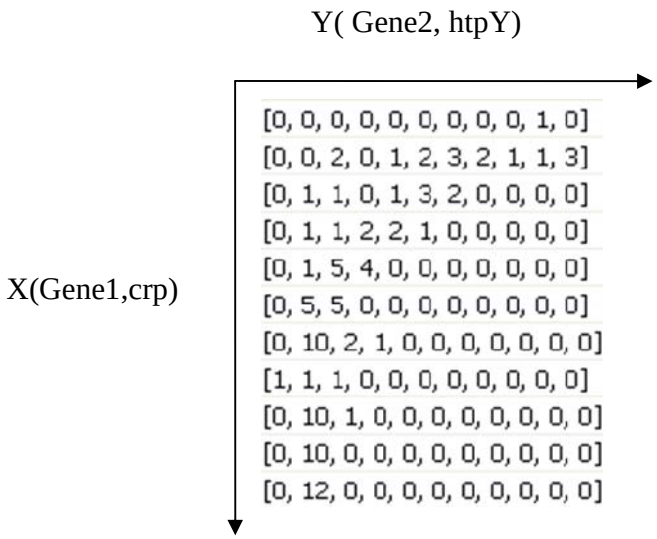


Figure 8-14 Example of two dimensional grid (bin Size=5) for crp and htpY

In Table 8-18 each feature (column) represents a cell and each row is a complete grid which belongs to a gene pair. We added a final column which was the indicator of the class label for that pair.

Table 8-18 An example of data transformation from the grid for feature weighting approach

X0 Y0	X1 Y0	X2 Y0	X3 Y0	X4 Y0	X5 Y0	X6 Y0	X7 Y0	X8 Y0	… .	X7Y 9	X8 Y9	X9 Y9	X0 Y1 0	X1 Y1 0	X2 Y1 0	X3 Y1 0	X4 Y1 0	X5 Y1 0	X6 Y1 0	X7 Y1 0	X8 Y1 0	X9 Y1 0	X1 0Y 10	Class
0	0	0	0	0	0	0	1	0	…	0	0	0	0	3	0	0	0	0	0	0	0	0	0	re

Algorithm: We applied a neural network algorithm specifically a multilayer perceptron with no hidden layer to find out the weight of each feature based on the class labels. The neural network that we used was the implementation of a multilayer perceptron from the Weka package (Hall, Frank et al. 2009). We used 10 fold cross validation resampling.

Result: This experiment confirmed our hypothesis. The accuracy was 68.67 percent. The result of the most important negative features for each class label is presented in the following table.

Table 8-19 The result of multilayer perceptron for feature weighting

”ac”		”re”	
Feature	**Weight**	**Feature**	**Weight**
x6y1	-12.78	x6y9	-9.15
x6y6	-11.27	x0y0	-6.89
x4y1	-9.07	x6y7	-6.18
x1y9	-9.04	x6y10	-5.85
x3y10	-8.9	x7y8	-5.7
x10y6	-8.47	x9y8	-5.22
x3y7	-8.26	x8y9	-5.22
x1y7	-7.48		
x1y8	-5.92		
x2y8	-5.92		
x2y10	-5.97		
x10y0	-5.2		

Table 8-19 shows two distinct areas with negative weights for “ac” and one for “re”. For “ac” the negative areas are the High-Low and for “re” are Low- High. This means features in High-Low area have negative weights in relation to “ac” class and features in the High-High area have negative weights in relation to “re” class. In other words, we can use the existence of the opposite relationship pattern in order to detect false positives. In this experiment, we aimed to find evidence from a data mining perspective to validate our heuristic post-processing. Here, we demonstrated that the concept of considering the absence of the opposite relationship for post-processing was supported by the expression data.

8.9 Experiment 9: Post-Processing Using Data Processing Inequality (DPI)

Purpose: In this experiment we applied an information processing technique called Data Processing Inequality (DPI) (Cover and Thomas 1991) to prune the search result to reduce the number of false positives. This method was previously applied in ARACNE (Margolin, Nemenman et al. 2006) along with Expected Mutual Information. The DPI procedure was applied to the output of our Fixed 2D Visualized Co-regulation function to reduce the number of its false positives and further improved the performance of pairwise discovery.

Setup: This experiment was performed on the output of our Fixed 2D Visualized Co-regulation function on the sixth dataset. We chose that output particularly as it had the highest number of true positives compared with other results. This provided us with a better opportunity to observe the effect of DPI in reducing the true positives as well as false positives. The DPI method was implemented in Python.

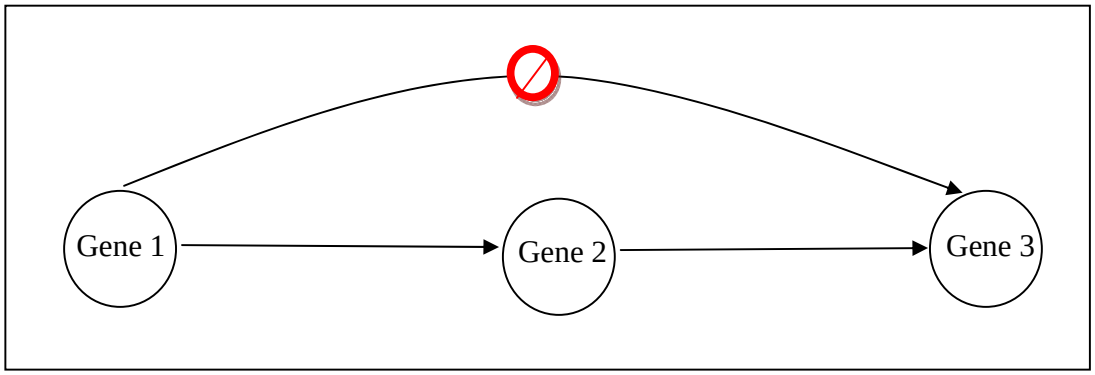


Figure 8-15 Example of Data Processing Inequality (DPI)

Algorithm: Consider we have a situation where there is a chain of three genes such as shown in Figure 8-16. Gene 1 affects Gene 2 and Gene 2 effects Gene 3. We eliminate the connection from Gene1 to Gene 3 if it is the weakest connection among the three of them. The connection is strong if it has a greater association value and is weak if it has a small association value. In other words if (gene 1, gene 2) > (gene 1, gene 3) and also

(gene 2, gene 3) > (gene 1, gene 3) then we can eliminate (gene 1, gene 3). Figure 8-16 Example of Data Processing Inequality (DPI)

The reason for doing this is that sometimes the observed relationship between genes is not a direct relationship and happens because of the influence of other genes in a chain. Association measures consider any relationships greater than a threshold as a direct relationship; however a strong correlation does not necessarily mean a direct relationship.

We were interested only to find direct relationships; therefore, we need a way to detect these indirect relationships and remove them. DPI did this job for us by detecting the weakest link in the chain and by assuming this weak link is the result of an indirect effect.

Result: The result of performing DPI on the output of our Fixed 2D Visualized Co-regulation function on the sixth benchmark is presented in Table 8-20. The sixth dataset had a higher number of true positives and this property enabled us to see the maximum effect of DPI on true positives. DPI process affects true positives as well as false positives. We also applied DPI on the fourth dataset and the result is presented in Table 8-21.

Table 8-20 The result of post-processing with DPI on the output of the fourth benchmark

	True Positives	False Negatives	False Positives	Recall/Precision	F-measure
Before	72	221	430	0.25/0.14	0.18
After	61	232	298	0.21/0.17	0.19

Table 8-21 The result of post-processing with DPI on the output of the sixth benchmark

	True Positives	False Negatives	False Positives	Recall/Precision	F-measure
Before	87	206	437	0.297/0.15	0.2
After	77	216	380	0.263/0.18	0.22

Significance Test: We performed a binomial test to find out if the results after and before applying the DPI post processing are significantly different. The following formulas show the calculation of the binomial tests.

Fourth Benchmark:

$$f(x,n,p) = \binom{n}{x} p^x q^{n-x} I_{(0,\ldots,n)(}x) \tag{8.4}$$

$$Before: y = \binom{641601}{72}\left(\frac{293}{641408}\right)^{72}\left(1-\frac{293}{641601}\right)^{641601-72}$$
$$= binopdf(X,N,P) = binopdf\left(72{,}723,\frac{293}{723}\right) = 2.12\, e^{-75}$$

$$After: y = f(x,n,p) : binopdf\,(61{,}591{,}293/591) = 5.9 * e^{-93}$$

Sixth Benchmark:

$$Before: y = f(x,n,p): binopdf(87{,}730{,}293/730) = 4.98 * e^{-64}$$

$$After: y = f(x,n,p): binopdf(77{,}673{,}293/673) = 7.5 * e^{-74}$$

The result confirmed the effectiveness of the DPI post-processing step; however we observed that by using DPI we removed some of true positives as well. This situation did not happen with our heuristic post-processing. We also observed that when the number of output records was higher the process was more effective.

8.10 Summary

This chapter mainly reported the experiments related to different versions of our proposed co-regulation function. Our association measure passed through three different

designs (versions). The first version was based on a one-dimensional bin. The function calculated a gene pair association in the following way. First, it discretized the values of each of gene into individual bins. Subsequently, it calculated the first quartile and the third quartile of each gene and considered them as the low and high boundaries. Then it counted the number of times that the two genes appeared as being High-High and High-Low in the same sample. Based on these numbers, our algorithm determined the strength and type of the relationship.

In the second version, we calculated a two dimensional grid for each gene pair. Then the area of interest was restricted using the first and the third quartile of each gene. The area of interest was the bottom right corner for “ac” (activation) and the top left corner for “re” (inhibition). If the numbers in the High-High area exceeded more than a threshold this relationship was labelled as “ac”. If the sum of numbers in the High-Low area exceeded more than a specific threshold, this was considered as an inhibition and the relationship was labelled as a “re”. A dual effect was observed when there was a considerable sum of High-Low and Low-High numbers.

This function demonstrated a high performance when it was compared with Pearson’s correlation measure and Mutual Information. We also showed by examples, that this function has the ability to show us any pattern that, by use of other measures, we would not be able to detect. This function also has a great visualization ability that can be used by experts in order to better understand the pattern of changes and the relationship between two genes.

In the third version of our association function, we tried to make the previous function a function with dynamic thresholds. In the first attempt, a sliding window was used over the grid to maximize a function and find the densest area inside the grid automatically. This algorithm was able to detect the area of interest dynamically; however its performance was not high enough.

In the second attempt, we used machine learning algorithms to learn the regulatory relationships patterns dynamically. In this way we could have a function with automatic

detection of thresholds. In doing this, we transformed the information inside the grid to a tabular format. This is the desired format for supervised classification algorithms. Each cell inside the grid became a feature and a row was made from all cells belonging to a grid. We used a combined dataset for this experiment which was a combination of our five benchmarks and also some artificial records. The artificial records were added to teach the clear regulatory relationships patterns to the classifier. We then applied a supervised classification algorithm (k-star) to this dataset. Our result demonstrated that we could train the machine learning algorithm to learn the regulatory relationships patterns. The result also showed that by increasing the number of "none" records; we could improve the classification performance.

We also tried to validate our assumptions about the regulatory relationship patterns in terms of machine learning by applying data mining techniques to our datasets. We transformed the data into a tabular format and applied a feature selection algorithm to discover the most important cells of the grid in relation to class labels. The result largely confirmed the validity of the regulatory relationship patterns we assumed based on the definitions of the regulatory relationships. In addition to feature selections, we tried another technique based on decision trees. In this technique, we tried to find general rules that could explain the relationships between the area of the densest cluster inside the gird and the class labels. In doing this, we applied the k-means clustering algorithm to discover the densest cluster inside the grid. Subsequently, we calculated the border of the densest cluster to find the minimum and maximum of boundaries. The above routine was performed on the known gene pairs which already had labels. Then, we used a decision tree to discover rules relating the location and density of the cluster to the class labels. This experiment also validated our assumption about the regulatory relationship patterns.

To achieve a higher performance, we decided to apply a post-processing step to eliminate some false positives. We applied two different post-processing methods. The first method was based on heuristics and the second one was Data Processing Inequality (DPI) a measure from information theory. Our proposed heuristic post-processing method looked for the absence of the opposite relationship in order to confirm a true

positive. Therefore, a false positive was defined where a pattern of the opposite relationship existed. For example if a pair was labelled as “ac” we looked for a pattern of “re”. If such a pattern existed we labelled this pair as a false positive and removed it. We limited this procedure to self-regulation only and achieved promising results. We also applied the DPI procedure to the result of our association function to delete false positives. This enabled us to further decrease the number of false positives. Both of the post-processing methods were effective and resulted in improvements in performance. The advantage of our heuristic post-processing over DPI was that our method was more accurate and did not remove any correct answers.

In summary, in this chapter we provided experiments with our co-regulation function. These experiments confirmed that using the definition of regulatory relationships in order to design a function, can improve the performance. We also provided supporting experiments that validated our assumptions about the regulatory patterns in terms of machine learning language. Our Fixed 2D Visualized Co-regulation function not only performed well but also provided us with a visualization tool. The visualization feature of our function has the potential to provide useful information for experts. We also showed by example that this function has the ability to show us any pattern that by use of other measures we would not be able to detect. Finally, we provided experiments related to post-processing procedures. Our heuristic based post-processing procedure was demonstrated to be quite useful and effective. We also applied DPI post-processing on the result of our co-regulation function and proved its effectiveness.

Chapter 9

Approach 3: Heuristics based on Network Structure

"See things not as they are but as they might be,"

- biography of Robert Oppenheimer

9.1 Introduction

The research question which drove this thesis was: *"How can reliance on microarray data and heuristics be reconciled to improve GRN discovery?"* In Chapter 6 we mentioned that the result of the second approach inspired us to use more domain knowledge. Thus in the third approach we incorporated more heuristic information compared with the other two approaches. In the first strand of the third approach, the definitions of the regulatory relationships were employed in order to design an effective association measure, which not only achieved a better performance but also had great visualization ability. We also employed heuristics to design a post-processing procedure as well as a background correction procedure. In Chapter 7 we introduced our co-regulation function and concepts behind that and in Chapter 8 we provided the experiments which supported the success of our idea.

In this chapter, we will introduce the second strand of the third approach. In this strand, the general question was the same, but we used another type of heuristic information to achieve an even higher performance. In the second strand, we used structural properties of the known networks in order to design an efficient computational model. The research question here was: *"How can we use the properties of known gene regulatory networks (such as structural properties) in order to design a more effective inference algorithm?"*

In this chapter we will describe our idea for using structural properties of the known network in order to design an effective GRN discovery algorithm. The idea is simple but powerful. We arrived at this idea by a comprehensive literature review and analysis of properties of biological networks, specifically gene networks. Firstly we will review the literature related to the structural properties and subsequently we will present a shortlist of the facts found in the literature.

In the second part of this chapter, we will discuss how we made use of such information in our modelling and we will present our algorithm based on such structural information. Finally, in section 9.5 we will report the results of our experiments with our designed algorithm and also the combination of this algorithm with the Fixed 2D Visualized Co-regulation function introduced in the previous chapter.

9.2 Analysis of Structural Properties of Biological Networks

In this section, we will review the structural properties of biological networks. The main focus is on gene networks and their properties. Our research question looks for any heuristic information that can be used to guide the search process in order to achieve a better performance. In order to obtain this information we explored the structural properties of the known biological network. As mentioned in Chapter 2 (Background), biological networks are modelled in bioinformatics as a connected graph. In this section we will describe some statistics and specifications of these biological graphs. Biological graphs can be divided based on the type of their elements to metabolic networks, protein interaction networks and gene transcription (regulatory) networks. In general, there are many similarities between these types in terms of the graph structure and properties.

The topologies of graphs are characterized by the degree of distribution, clustering coefficient, and with the presence or absence of subgraphs or motif characteristics (Almaas, Vazquez et al. 2007). There are also other characteristics and properties of a

graph that cannot be obtained from topological information and we will review them separately.

9.2.1 Degree Distribution (Scale-Free Property)

In general, biological networks are far from fully connected. In a fully connected network with K nodes, the number of edges is K^2. Experiments have demonstrated that in the biological networks the number of edges is approximately K (Kepes 2007; Maslov 2007).

This shows that this graph is a scale-free graph. Scale-free means that when there is an increase in the number of nodes the number of edges does not scale up as the square of the nodes but instead increases with the same order as the nodes. As a result, a scale-free graph has a specific topology in which some nodes act as highly connected nodes having high degrees, while most other nodes are of low degree.

The degree of a node is the mean number of edges connected to the node. In other words, in a scale-free network some nodes (known as hubs) are highly connected and most of the nodes are less connected. Scale-free networks' structures and dynamics are independent of the number of their nodes (Kleinberg 2000). This means a network that is scale-free will have the same properties regardless of the number of nodes. Many real world networks are scale-free networks which makes the study of this class of networks important. Formally, scale-free networks show the distribution illustrated in Equation (9.1), known as a power law relationship (Barabási, Newman et al. 2006; Junker and Schreiber 2008).

$$P(k) \sim k^{-y} \tag{9.1}$$

Figure 9-1 represents Equation (9.1). In this figure and equation, *k* (horizontal axis) is the average node degree, *y* is a parameter whose value is typically in the range $2 < y < 3$ and the diagram represents the fact that there is a reverse relationship between *k* and the

degree distribution *P(k)* (vertical axis). This means that when degree increases the degree distribution decreases.

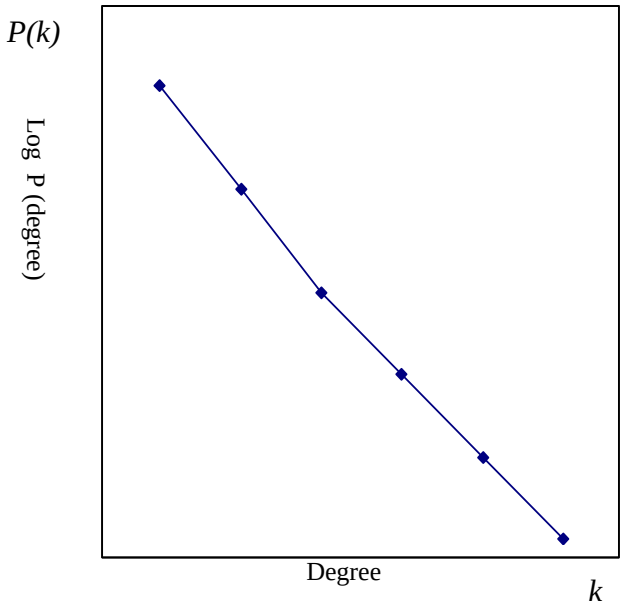


Figure 9-1 Characterizing degree distribution of biological networks

Figure 9-2 presents a protein interaction network which is an example of scale-free networks in biology. As you can see, Figure 9-1 represents a specific structure known as scale-free structure where a few nodes are highly connected and the majority of nodes are connected to the average of two other nodes. Different colours represent different types of proteins.

Biochemical activity in both metabolic and genetic networks is dominated by several *Hot links* that represent a few highly active nodes embedded into a web of less active nodes. This is not a unique feature of biological systems: “Hot links” appear in a wide range of non biological networks where the activity of the links follows a wide distribution. The root of this property comes back to the network topology; the “scale-free” nature of the network (Menezes and Barabási 2004). The highly connected hubs seem to act as glue in these webs, allowing most nodes to connect to each other through a small number of jumps. In a random network only 27% of the nodes are reached by the five most

connected ones, while we reach more than 60% of the nodes in a scale-free network. This illustrates the key role played by the hubs. Hubs are also the most important and the most essential genes (Goh, Cusick et al. 2007). In regulatory networks, hubs form an important centre of regulation with a far-reaching control. They have significantly more outgoing edges than incoming edges with many feed forward loops around them to make them robust against failure (Konagurthu and Lesk 2008; Seo, Kim et al. 2009).

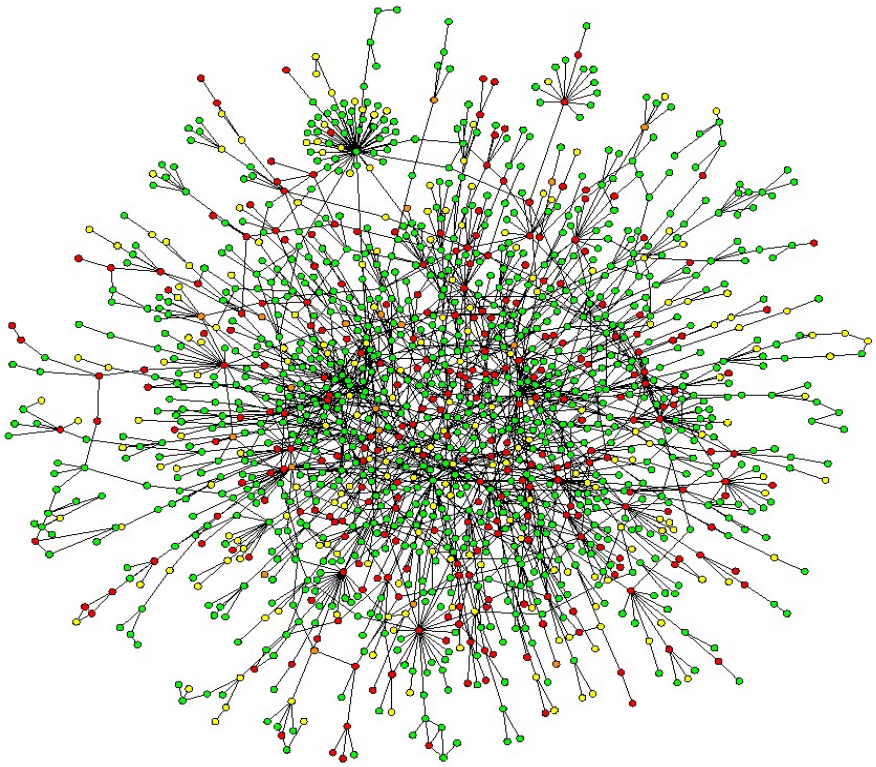


Figure 9-2 Yeast protein network interaction (from http://www.bordalierinstitute.com/images/yeastProteinInteractionNetwork.jpg)

9.2.2 Average Path Lengths (Small World Property)

Biological networks have small average link distance between any two nodes (small-world property)(Albert 2005). In small world networks most of the nodes are not neighbours of each other, but can be reached from every other node by a small number of

edges or steps. In other words they are rich in local connections, with a few long range connections (Kleinberg 2000). This sort of graph can be classified based on the mean-shortest path length. Small world networks look like some dense subgraphs which are connected through a few edges to make the overall network (Kleinberg 2000; Vázqueza, Flamminia et al. 2003). Many real world problems have the small network property, such as the Internet and social networks, as well as gene and protein networks. The majority of genes in all databases have less than ten interaction partners (Mathivanan, Periaswamy et al. 2006). The average number of links or degrees reported by different studies vary, but fall between two and seven (Hallinan and Wipat 2006; Mathivanan, Periaswamy et al. 2006).

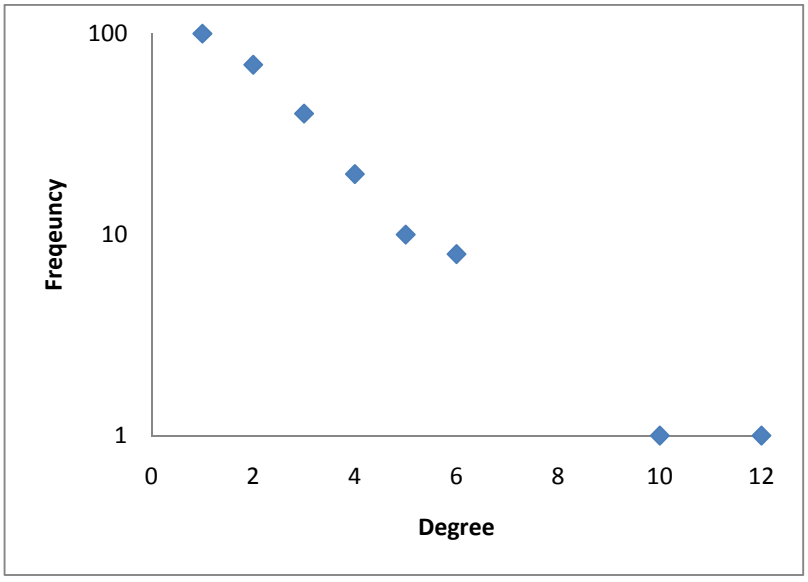


Figure 9-3 The relationship between degree distribution and percentage of genes based on (Davierwala, Haynes et al. 2005)

Another feature of biological networks is that the most important elements (more essential for our body) have a higher degree (Davierwala, Haynes et al. 2005; Roth 2005). For example, the highly connected proteins are more essential and their average connectivity is around 10-15 edges per node. This means hubs which are the most connected nodes produce the most important and essential proteins (Goh, Cusick et al. 2007).

9.2.3 Clustering Coefficient (Modularity)

A common technique for measuring the tendency of an element to form a cluster in a network involves determining the clustering coefficient. Clustering coefficient (CC) is defined as the edge density around a vertex's neighbours and can give us insight into the local structure of a network. Clustering coefficient is also known as clustering of a node, and is defined for a node i with degree K_i as follows (Almaas, Vazquez et al. 2007):

$$C_i = \frac{2\, n_i}{K_i(K_i - 1)} \tag{9.2}$$

The clustering coefficient is calculated for each node and then overall clustering is given by $\sum \frac{C_i}{N}$. For example, in a small world protein interaction network, a high clustering coefficient property indicates that proteins are likely to form a subnetwork (clique) or a dense cluster by their interactions. This suggests a structure where some dense clusters are connected to each other through links and most of the edges have a very short alternative path length. Only less than 5% of the edges have the shortest alternative path length of more than five (Pei and Zhang 2005).

It has been shown that for biological networks the average clustering follows a power-law form as $C(k) \sim K^{-\alpha}$. This suggests an existence of a hierarchy of nodes with different degrees of modularity (measured by the clustering coefficient) overlapping in an iterative manner (Ravasz, Somera et al. 2002; Almaas, Vazquez et al. 2007). These modules are usually functional modules which are a group of genes working together to perform a function.

Biological networks are modular and these modules are associated with well-defined functions as you can see this in Figure 9-4. Modules have been defined as functionally buffered, robust, independently controlled, plastic in composition and interconnectivity, and evolutionarily conserved. The evolutionary conservation of modules was especially beneficial for gene networks involved in early development. This special feature of some of the modules is tightly linked to their robustness under different sources of noise.

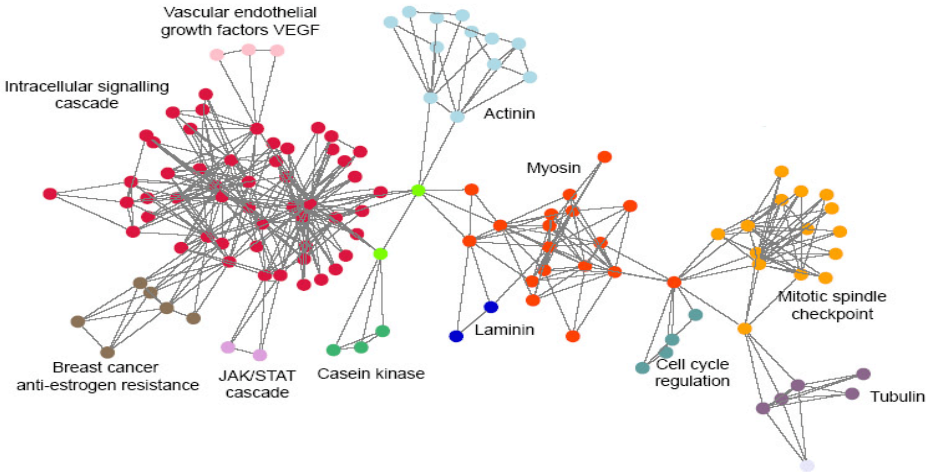


Figure 9-4 A Protein Interaction Network (from: http://science.cancerresearchuk.org/sci/lightmicro/116743)

9.2.4 Motifs and Subgraphs

We described earlier in the Introduction and Background chapter that a gene can activate or inhibit another gene. A dual effect is possible as well, when in some circumstances there is an activation effect and sometimes there is an inhibition effect (Kepes 2007). This implies that a graph of genetic interactions is naturally of a directed graph.

The interaction between genes is not limited to pairs. A gene might activate another gene which in turns causes a third gene to be activated. Network motifs are small building blocks composed of two or more interactions (or 'edges') that are overrepresented in GRNs compared to randomized networks (Milo, Shen-Orr et al. 2002). The interaction between genes causes a cascade or genetic regulatory pathway. If such a pathway is closed onto itself, it forms a feedback circuit. Some examples of feedback circuits are self-regulation and feed-forward loops. The best-studied motifs are feed-forward loops. Transcriptional regulatory networks of cells and some electronic circuits are all information processing networks that contain a significant number of feed-forward loops. This pattern is found in GRNs from Bacteria, yeast and C. elegans (Arda and J.M.Walhou 2009). This pattern does not show up in the other type of networks. In

transcriptional regulatory networks, this pattern melds into patterns of clusters (Almaas, Vazquez et al. 2007). Of course there is a global feedback in the network as well.

Another important motif is self-regulation. It has been demonstrated that as many as 59% of the transcription factors in Escherichia coli regulate the transcription rate of their own genes (Hermsen, Ursem et al. 2010).

9.2.5 Other Properties

In addition to the above properties, there are some other structural properties that we did not discuss above, but which have been identified in system biology literature. The following information is from Kitano (2007).

1. Nature uses network in order to make a system which is robust against noise and failure.
2. Positive feedbacks are used to create bistability and make the system stable against minor perturbation in stimuli and rate constants.
3. Alternative mechanisms and concepts for a function increase tolerance against failure.
4. Modularity provides isolation of perturbation from the rest of the system. A cell is the most significant example. Modules inside the network buffer the perturbations.
5. Buffering properties isolate noise and fluctuations (decoupling).

9.3 Summary of Heuristics Based on Structural Properties

In summary, there is strong evidence that biological networks are scale-free, small world, hierarchical and modular (Ravasz, Somera et al. 2002; Ma, Kumar et al. 2004). They also show specific patterns. They have a significant number of self-regulatory elements and feed-forward loop (59% in E. coli).

Network models therefore have the following properties:

1. Nature uses networks in order to make a robust system against noise and failure.
2. Biological networks are represented as connected graphs in the computational world.
3. They demonstrate scale-free network properties with hierarchical structure.
4. There is a small average link distance between any two nodes (small-world) property.
5. The average degree falls between 2 and 7 (average around 5).
6. Highly connected nodes called *hubs* are more essential, having a degree of 10-15. They provide a strong backbone for the network and any mutation of them is critical and fatal (Barabási, Gulbahce et al. 2011).
7. GRN networks have a considerable amount of loops and especially have a motif of feed forward loops which are melded into the structure of the network.
8. Positive feedback is used to create bistability, which make the system learn against minor perturbation in stimuli and rate constants.
9. 59% of the total transcription factors in Escherichia coli regulate the transcription rate of their own genes (Hermsen, Ursem et al. 2010).
10. Alternative mechanisms and concepts for a function increase tolerance against failure.
11. Modularity provides isolation of perturbation from the rest of the system. A cell is the most significant example. Modules inside the network buffer the perturbations.
12. Buffering properties isolate noise and fluctuations (decoupling).
13. Bow-tie global architecture (diverse overlapping inputs and output cascades) is connected by a core network (Zhao, Yu et al. 2006).
14. Indirect relationship is where there is a chain between two genes and it usually shows a weaker relationship. The longer the chain, the weaker the interaction.

The direct relationship presents a strong gradual change (Akitaya, Seno et al. 2007).

15. The distribution of the pairwise correlation coefficients of genes follows a power law. That is, while the majority of gene pairs have only a few links, a few gene pairs display a significant number of edges connected to them (Kepes 2007).

The above information about properties of gene networks can help us to choose the best representation for GRNs and design efficient algorithms for GRN inference. In this thesis we used structural properties of the network in our second and third approaches. In the second approach we used local properties of the network and its modularity to design a more effective global search algorithm based on a Genetic Algorithm. Our solutions in the genetic algorithm were partial solutions, each containing a subgraph and at the end of the evolutionary process we merged them to find the complete network. Our fitness function also considered the shape of the network to favour the solutions which were more similar to the known networks with small world property.

In the first strand of the third approach, we mainly used the definition of regulatory relationships, which is not a structural property. We also used fact number 14 (about indirect relationships) to find the indirect relationships and eliminate them. Here in the second strand of the third approach we used more structural properties in order to achieve a better discovery process. The main property that we used was the scale-free property. Specifically, we used the concept of hubs and their relationships. In the first step, we found the hubs then we built a network of hubs in order to build the core structure of the unknown network. Our algorithm builds the structure of the network in an incremental fashion by taking into account structural properties such as the degree of each node and shortest path length. The detailed description of our method is presented in the following section.

9.4 Approach 3: Algorithm Based on Heuristics from Network Structure

Regardless of the approach for reverse-modelling of GRN the hardest point is to identify the network structure (d'Alche-Buc 2007). This problem is known to be a NP-hard problem (Chickering, Heckerman et al. 2004). NP-hardness either calls for a relaxation of the combinatorial problem into a continuous one or demands heuristics to explore the finite but huge set of candidate networks for a given number of genes. We looked at the network properties to find which heuristics we could use in order to find the structure of the graph more effectively. The key finding was Hub Network.

Hubs are the most important elements in the network. They are usually the most important genes which regulate many other genes. Therefore any mutation or changes in them results in the collapse of the network structure. Hubs are evident in many laboratories experiments because they have so many connections and are therefore well known genes. Further investigation told us that the hubs are connected. They make a network which is an essential and critical part of the GRN. This primary structure, the network of hubs, acts as a backbone to keep the network together. Figure 9-5 shows the hub nodes in the full E. coli network (Ma, Kumar et al. 2004) which we drew using Cytoscape software (Shannon, Markiel et al. 2003). The Hub Network which builds the core structure of the network is highlighted in the graph.

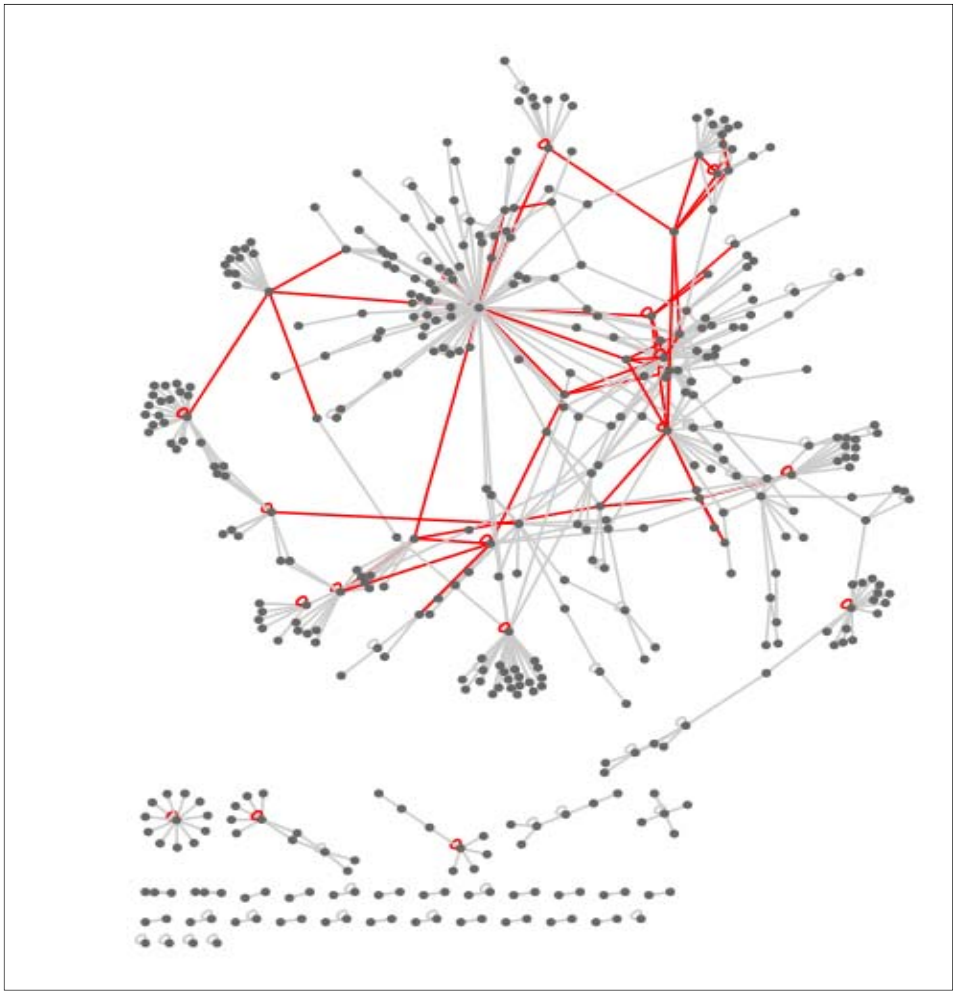


Figure 9-5 The E. coli full regulatory network and its Hub Network

This property of the network gave us the idea to use the Hub Network in order to build the core structure of the unknown network. It was not desirable to be limited to existing knowledge as this would limit the discovery of new information therefore we needed to consider not only building the network based on the connections in the known network, but also based on what the expression data shows. Despite the changes in the number of connections between normal and diseased conditions, the overall distribution remains the same. There is evidence that differential connectivity follows systematic gain or loss of connections and distribution of gain and loss is symmetric (Leonardson, Zhu et al. 2010). Therefore we can assume that connections in experimental conditions follow the same

distribution as the known network. In addition, it was demonstrated that nodes with high connectivity (hubs and super hubs) tend to have low levels of change in gene expression and genes with a high level of change in expression are more likely to be peripheral nodes (low connectivity) in the network (Lu, Jain et al. 2007).

Finding the structure of a GRN is both the most difficult part and the initial part of GRN discovery. Therefore finding the right structure will help improve the process drastically. By using a network of hubs we have the opportunity to not only find the right structure, but also to find the network which has similarity in terms of structure with the networks from domain knowledge.

There are further heuristics from the domain that we used here. Domain knowledge can tell us the average degree of each hub. This knowledge usually is in the form of known interactions saved in a public database and based on such knowledge; we usually know approximately how many genes are connected to each hub. This can guide us to where we want to add nodes to the Hub Network. The network structure also has small world property; therefore the average neighbourhood size is around 5.

In order to use such information we decided to build the network incrementally. We decided to build the first layer or fundamental structure of the network from hubs by looking at the known network. We also considered building the connections between hubs based on the expression data. This resulted in a network of hubs which makes the first layer of our target network.

The second layer was calculated based on the first layer. The second layer was built by the most connected nodes to the hubs. Having known the degree of each hub and having known that non hubs have average degree of one or two; we applied a background correction after computing the pairwise association of each gene by considering these facts.

In our background correction process, we first normalized the correlation values between each gene and any other genes and filtered those which were less than 0.5. Then we chose the top one from the list according to the degree of the node and attached them to

the node. We created a rule of thumb for this selection procedure. If the degree of the node was less than 15 we chose exactly according to the degree of the node, otherwise we chose 15 plus one third of the degree minus 15.

$$n = \begin{cases} Degree, & degree < 15 \\ 15 + (Degree - 15) * 0.3, & degree \geq 15 \end{cases} \quad (9.3)$$

In (9.3), *n* is the number of genes which we chose from the normalized ranked list. This equation tells us that if a hub node has degree of 10 in the domain knowledge then we chose 10 nodes from that list to be attached to it. If a hub's degree is 42 (like "crp") we chose only 15+ (42-15)/3 which is 24. As mentioned in Chapter 8 , this helps to eliminate connections which are the result of genes that appear to have connections with too many other genes based on microarray data.

In the background correction process all non hub genes get only their top ranked gene attached to them (as they have degree of one). Based on the information from the literature non hubs have usually less than five nodes and are most likely to have one or two connections. Therefore we chose to attach only one node to non hub nodes to build the third layer of the network.

We also built the network incrementally layer by layer. There is literature supporting such an incremental approach using a dynamic threshold. In a study by Gowda et al. (2009), the correlation for each gene with the rest of the genes is calculated and sorted based on their rank, then a classification method was used to decide whether to add a connection between genes from the ranked list (starting from top ones) to the given gene, whenever the error of classification does not increase.

We also analysed the skewness of the expression distributions in order to find whether there is a relationship between false positives and the amount of skewness of distribution. We found that there is a relationship between skewness of more than a certain threshold and false positive answers. Therefore we pruned the answers related to genes with extreme skewed distributions, if skewness of the gene was greater than 1.5 or the sum of the skewness of the two genes in the pair was greater than 2.2.

The proposed algorithm takes into account all of the above processes and is as follows:

1. Find the hubs from the known network (We chose any node with degree more than four to be considered as la hub).
   Also find the degree of each hub.
2. Calculate pairwise associations between each pair of hubs in order to find the network of hubs. The pairwise associations are calculated based on the expression data only. For this purpose any association measure can be used such as Pearson's correlation or our proposed co-regulation function.
3. Now that the Hub Network is established, build the second layer by finding more nodes and adding them to the network incrementally. In doing this, calculate pairwise associations for each hub against all other non-hubs.
4. Remove those genes which show a skewed expression distribution (This is because those genes with extreme skewed distributions are more likely to be noise).
5. Normalize these association values for each hub and then rank them and choose the top ranked one according to the formula presented in (9.3). Add those top ranked ones to the Hub Network. Now the second layer of the network is completed.
6. Calculate the pairwise association for each node in the second layer with every other node which is not already in the network. Normalize the values and choose the most correlated gene to add to the network (considering degree of two for non hubs). Consider removing the skewed one here as well. Now the third layer of the network is ready.
7. For those nodes that are not already in the network, calculate the most correlated place to put them.

We have to mention that the result of this algorithm using the co-regulation function is a directed network as our co-regulation function specifies a directional regulatory relationship between each pair of genes.

9.5 Experiment 10: Using Hubs for Structure Discovery

Purpose: To test how effective the algorithm to find the core structure of the target network.

Setup: In this experiment we used the default dataset (set 1). We also used the original E. coli network to find the hubs and their degree.

Algorithm: As illustrated in the previous section above.

Result: The result of this procedure gave us 100 percent accuracy in the first layer, where the Hub Network matched exactly. The accuracy for the second layer and the last layer dropped considerably. The target network was also quite similar to the known networks in terms of the structural properties. The result was clearly promising and demonstrated the great ability of using hubs to build the GRN network. Table 9-1 and Figure 9-6 show the Hub Network discovered in the first stage of our algorithm which was 100% matched with the Hub Network from the E. coli full regulatory network. The last column in the table indicates the measure of co-regulation between the two genes. We tried Pearson's correlation along with this approach as well, but the presented result here came from our 2D Visualized Co-regulation function. In that function, any number greater than 0.17 is considered as a significant association and the maximum association was 0.25 for "ac" and "re".

Table 9-1 Hub pairs extracted in Experiment 11

Gene1	Gene2	Relationship	Co-regulation
crp	dcuB_fumB	re	0.18
fnr	dcuB_fumB	ac	0.17
narL	dcuB_fumB	re	0.15
arcA	focA_pflB	ac	0.19
crp	focA_pflB	ac	0.19
fnr	focA_pflB	ac	0.2
himA	focA_pflB	ac	0.2
narL	focA_pflB	re	0.17
arcA	nuoABCEFGHIJKLMN	re	0.21
fnr	nuoABCEFGHIJKLMN	re	0.21
himA	nuoABCEFGHIJKLMN	re	0.21
narL	nuoABCEFGHIJKLMN	ac	0.18
crp	rpoH	du	0.17
cytR	rpoH	re	0.18
rpoE_rseABC	rpoH	ac	0.2
arcA	Soda	re	0.17
fur	Soda	re	0.18
himA	Soda	re	0.19
crp	tdcABCDEFG	ac	0.17
fnr	tdcABCDEFG	ac	0.18
himA	tdcABCDEFG	ac	0.18

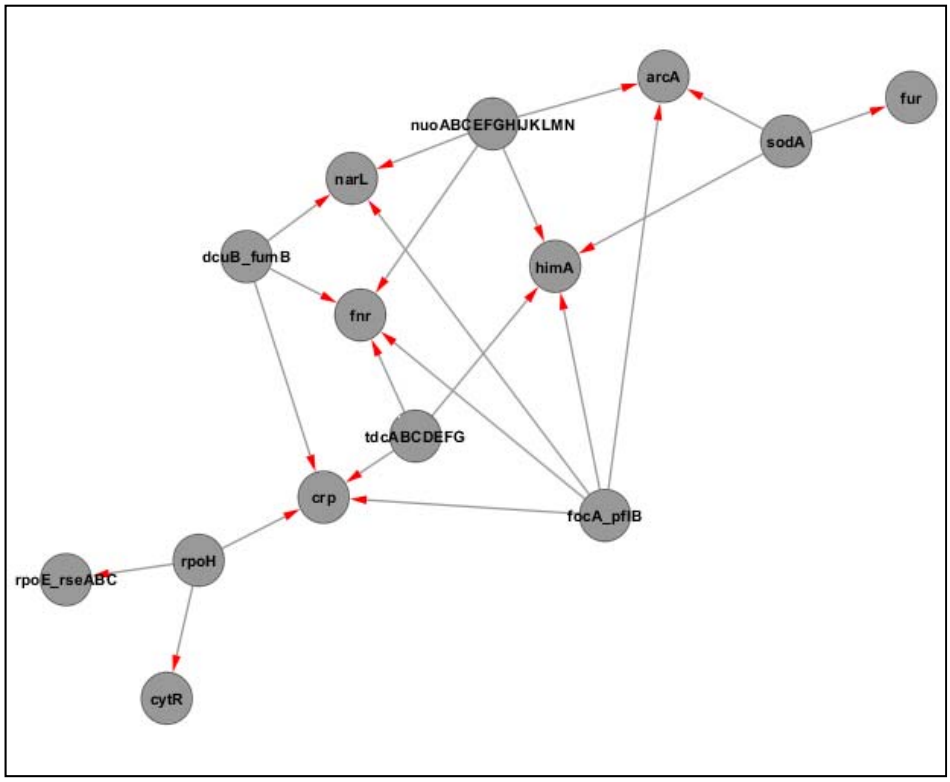


Figure 9-6 The Hub Network generated in Experiment 11

9.6 Experiment 11: Using Hubs along with 2D Co-regulation Function

Purpose: To test the maximum performance of the model that employs the Hub Network and Fixed 2D Visualized Co-regulation function. The purpose of this experiment was to make further improvement in the performance of GRN discovery by using all the elements of the third approach. In this experiment we did not use only the DPI method for post-processing as it was tested later on after this experiment.

Setup: This experiment was performed on the first and the second dataset because they have two different network structures. In this experiment we used our Fixed 2D Visualized Co-regulation function along with the Hub Network algorithm.

Algorithm: The algorithm is similar to the previous one, the only difference between this experiment and the previous experiment is that this one was applied to the output of our Fixed 2D Visualized Co-regulation function, after it passed the heuristic post-processing step.

We used the 2D Fixed Visualized Co-regulation function described in section 8.3 and our heuristic post-processing described in section 8.8 for this experiment. Basically our Hub Network algorithm, mentioned previously, can be applied to the output of any other association function, but for reaching the maximum performance we used our 2D Fixed Visualized Co-regulation function here.

We first applied our association function to expression data and chose the final pairs and then applied the heuristic post-processing step for further reduction of false positives. Then we used those final pairs as the input for our Hub Network algorithm.

Result: We compared the performance of our system against some of the most well known systems for GRN discovery. For this purpose, we used implementation of these systems in *minet* package (Meyer, Lafitte et al. 2008) inside Bioconductor. We called them through our Python code using RPy scripts.

Table 9-2 Result of Experiment 12: Using hubs along with our co-regulation function on the first benchmark

Method	Dataset	Total Records	True Positive	False Negative	Recall	Precision	F-measure
ARACNE	1	140	24	269	0.082	0.17	0.11
CLR T=10	1	589	39	254	0.13	0.07	0.09
Hub Network	1	206	41	252	0.20	0.14	0.17

Table 9-3 Result of Experiment 12: Using hubs along with our co-regulation function on the second benchmark

Method	Dataset	Total Records	True Positive	False Negative	Recall	Precision	F-measure
ARACNE	2	218	31	203	0.13	0.14	0.14
CLR T=8	2	842	64	170	0.27	0.08	0.12
mrnet & norm=0.5	2	178	28	206	0.119	0.15	0.14
Hub Network	2	221	47	187	0.22	0.21	0.21

We ran ARACNE (Margolin, Nemenman et al. 2006), *mrnet* (Meyer, Kontos et al. 2007), CLR (Faith, Hayete et al. 2007) on the first two benchmarks and the results are presented in Table 9-2 and Table 9-3. We can see that the result improved even further compared with applying solely the 2D Visualized Co-regulation function. This result reported further improvement on the performance, especially over the second benchmark. The difference between the performance of our system and other systems was more considerable on the second benchmark. The reason for that is because the second dataset had a similar structure to the known networks compared with the first one. Therefore our system which employed structural information of the known networks resulted in a better performance on the second benchmark. Figure 9-7 shows a partial view of the network produced by our system using the second benchmark.

Significance Test: We ran a paired t-test on the above results to find if the results from the HubNework algorithm are significantly better than those of ARACNE and CLR. We chose paired t-test as the changes in F-measure, precision and recall for different methods are not independent. As paired t-test assumes that there is no difference between the variance of two distributions, we ran F-test to confirm this assumption.

Table 9-4 Result of significance test for Experiment12

Method	Recall	P-value	Is Significant
ARACNE against HubNetwork	=TTEST({0.11,0.14},{0.17,0.21},1,1)	0.0244	Yes
CLR against HubNetwork	=TTEST({0.09,0.12},{0.17,0.21},1,1)	0.0187	Yes

This result confirms the effectiveness of our Hub Network heuristic. Not only did the system find the interactions that existed between hubs (existing knowledge) as we showed on the previous experiment, but also it was capable of finding new interactions as was indicated by the rate of true positives while producing far less false positives. We have to mention one characteristic of the Hub Network algorithm here. In this algorithm, the size of the output network does not vary as much as in the other algorithms. The reason behind this is that the structural information is constant and the only difference relies on the Co-regulation function performance therefore, it shows a smoother behaviour. Using the structural information has both advantages and disadvantages. The main advantage is improved performance and the disadvantage is that it makes the approach dependent to the availability of the information about known hubs from domain knowledge. In the absence of such information the first step cannot be applied but the rest of the steps of the algorithm are still applicable.

As mentioned earlier our Hub Network algorithm can be used in combination with other association functions such as Mutual Information. It can also be employed in Bayesian Network approaches.

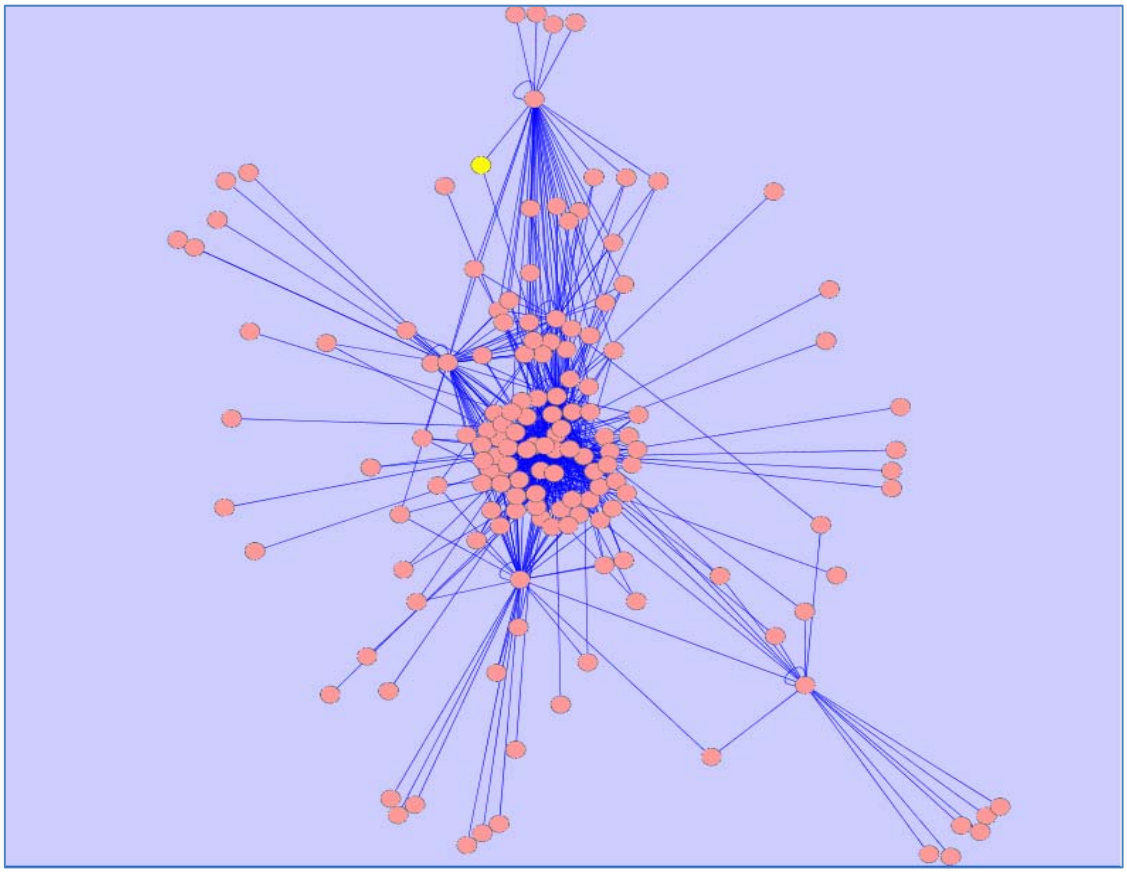


Figure 9-7 Visualization of the result network produced in Experiment 12 using Cytoscape

9.7 Summary

In this chapter, we introduced and investigated the second main idea inside our third approach. In this approach, we used more heuristics from molecular biology domain knowledge to improve the process of GRN discovery. Specifically, we used the Hub Network to construct the primary structure of the network. We also used the degree of each hub in our process.

We first explained our overall algorithm, and then provided the related experiments. Our algorithm first extracted hubs from the known GRNs related to the same organism to build a network of hubs based on microarray expression data. This provided us with the first layer or foundation of the target network. We used this small network as a primary

structure of the new network to build the rest of the network. The algorithm also used the degree of each node to build the network incrementally layer by layer. The degrees of nodes were extracted from the domain knowledge and gave us a clue about possible connections in the unknown network. Based on this information, we created a procedure which based on we determined how many nodes we needed to attach to the hubs and also the non hubs in each layer.

In the second part of the chapter, we presented experiments related to our final system. In our system we combined three algorithms. We combined our Fixed 2D Visualized Co-regulation function to identify edges (gene pairs), our post-processing procedure to decrease the false positives rate, and our Hub Network algorithm to identify the correct structure of the target network. All major ideas of our third approach were combined in this system to achieve the best performance. We provided the result of our system's performance on two benchmark datasets.

The result reported improvement not only in the number of true edges detected, but also in the decrease of false positive edges compared to some other methods. One interesting characteristic of our method compared with some other methods was observed when we visualized our output. Our output GRN showed a similar structure with the known networks compared with the other methods. This property is lacking in other methods which do not consider domain knowledge. This demonstrates the great benefit of using hubs to build the primary structure of the GRNs.

Chapter 10

Conclusion and Future Directions

"Just a moment, I've almost finished' If on a Winter's Night a Traveler'."

-Italo Calvino, If on a Winter's Night a Traveler (1979)

10.1 Conclusion

Biological systems have traditionally been studied by focusing on individual cellular components. Though the knowledge gained in this way is insightful, it has been increasingly clear that the understanding of complex cellular systems requires understanding of how different components work together. The advent of high-throughput technology like microarrays, where cellular activities can be measured on a genome-wide scale, has provided opportunities to obtain a systematic view which is known as system biology. The advances in system biology, which look at the system level of cellular activities, have an innate ability to achieve the objective of prognosis of disease and potential drugs/treatment development.

DNA microarray technology provides us with a picture of the whole genome at once. In this way we can systematically study the relationships between genes in a certain condition which can then be represented as a Gene Regulatory Network. To achieve this goal we need to obtain several samples during developmental stages of a condition. A condition can be an artificial perturbation such as gene knocking-out or the effect of an environmental factor such as dehydration. The goal is to find the network of genes responding to that condition.

Microarray technology gives us a facility to build a gene regulatory network. However, analysing such data is challenging due to its high dimensionality, noise and various kinds

of biases, which reduce the accuracy of knowledge discovery techniques applied for the aforementioned applications.

There are various techniques and modelling approaches towards the problem of gene regulatory network discovery, ranging from differential equations, which provide detail of interactions, to directed graphs representing only a topological model.

In order to achieve the objective of gene regulatory network discovery in this thesis, we applied different modelling techniques inside three approaches. Our initial aim was *"How can we improve gene regulatory network inference?"*. By understanding the benefit of using domain information we then decided to focus on the question of *"How can reliance on microarray data and heuristics be reconciled to improve GRN discovery?"*.

To address our research question, we presented three novel computational approaches. In the first approach we did not use any information from the domain and tried to increase the performance of GRN discovery by creating a better computational technique. Our novel computational technique achieved a higher performance compared with other methods, but still this approach was not scalable enough. Then we moved to the second approach which incorporated some heuristics to navigate the search space effectively. The second approach was applicable for a genome-wide GRN but the performance was not good enough. Finally, the third approach used the highest amount of heuristic information. The third approach successfully outperformed some of the best systems for GRN discovery and was the most successful. In this thesis we have shown using heuristics can improve the GRN discovery process.

In the remainder of this thesis, we will summarize the main contributions of each approach and then conclude with a discussion of directions for the future work.

10.2 Summary of Results and Contributions

10.2.1 Approach 1: Memetic Gene Expression Programming

In this approach we used a system of differential equations to model the problem. The question was: *"How can we improve the performance of current techniques applied to differential equations modelling of GRNs?"* To answer this question we proposed using Gene Expression Programming combined with a local search mechanism to improve the regression procedure, aiming to find a system of differential equations. The combination of the local search methods with evolutionary algorithms is known as Memetic Algorithms (Moscato and Norman 1992). Memetic algorithms are more efficient than GAs and are also more scalable. This is because of the use of local search which enable the algorithm to reach the optimal solution faster and more accurately (Hart, Krasnogor et al. 2005). By combining a local search with Gene Expression Programming (Ferreira 2001) we expected to increase the performance of GEP. The proposed approach advanced the ability of Genetic Expression Programming in this problem. Our technique surpassed the conventional genetic programming performance by a factor of one hundred in terms of the quality of the final solution. It also improved the ability of gene expression programming in finding constant values and parameters. The combination of a local search method with Gene Expression Programming is proposed for the first time in this thesis. The proposed combination proved to be effective and is promising to be effective in other applications as well.

Based on this study, it is recommended that the Memetic Gene Expression Programming technique is a useful technique for solving a system of differential equations. We particularly demonstrated its effectiveness for solving a system of differential equations used to model a GRN. We also recommend that differential equation modelling is not applicable for a large scale real size network.

Specific Computational and Bioinformatics Contributions

Our contribution to Computer Science:

1. Introducing a new method called *Memetic Gene Expression Programming* which surpassed the performance of current evolutionary techniques for finding a system of differential equations.

Our contribution to the Bioinformatics:

2. Applying the *Memetic Gene Expression Programming* technique to a GRN discovery problem modelled by a system of differential equations.

10.2.2 Approach 2: Combined Evolutionary Algorithms (Memetic Algorithm)

The second approach ignored details of quantitative modelling and focused on finding the big picture of the GRN structure to be able to discover genome-wide GRNs. To achieve this aim, this approach introduced the use of heuristics based on domain knowledge in combination with a genetic algorithm. In this approach, for the first time, we combined a combinatorial search method with the domain knowledge for GRN discovery. The domain knowledge was used in the form of a local search process to make the solutions similar to domain knowledge. In this way, we introduced a new design for GA and also a new memetic algorithm. The new GA was different in the way that we designed a chromosome as a partial solution in the form of a subgraph. The new MA was different as it used a case retrieval mechanism to retrieve similar cases from the domain knowledge to replace the chromosomes (subgraphs) with these cases. In contrast with the existing MAs which employ a local search method, such as hill climbing which need to capture information in the form of a function, in our proposal we did not use any function and instead, the cases from the domain were used directly. We called this new local search *cultural imitation*.

Our attempt to design a combined evolutionary algorithm using information from domain knowledge showed us that a simple GA works well when the search space is big and

coding is simple. However in this application, we could not use the simple GA to find GRN structure, as that required a huge chromosome which was hard to evolve in order to find the solution. That is why we tried to build the whole GRN network by evolving its smallest elements (functional gene subnetworks). This also gave us a facility to incorporate the domain knowledge which is in the form of gene subsets and subnetworks. The GA mechanism suffered from early convergence therefore the result was not comparable.

The lessons from the second approach taught us that we needed a simpler search mechanism to explore the search space and GA was not an effective tool for this purpose. By using considerable information from domain knowledge, the search space is not as large. Thus, in the third approach we tried to use as much heuristics that we could to limit the search space and build an effective model.

Specific Computational and Bioinformatics Contributions

Our contribution to Computer Science:

3. Exploring a new GA algorithm design in which chromosomes are the partial solutions in the form of subgraphs and we evolve these partial solutions to find the complete solutions.
4. Exploring a new memetic algorithm design. Our memetic algorithm suggests a new concept of meme. In our design, local search played a role of imitating from nature and we used a case retrieval mechanism to make the solutions similar to one in the domain knowledge.

Our contribution to Bioinformatics:

5. Combining information about functional gene sets inside a combinatorial search process for the first time.

10.2.3 Approach 3: Heuristics Based Association Measures

The third approach aimed to use more heuristics and reported the best results. This approach proposed two major new ideas and several minor ones. The first major idea

used heuristics to define an association measure. The contributions in that section are as follows:

a. The first and most important idea was using the definition of regulatory relationships to compare and validate different association measures. As a result of this study we found that existing association measures are not designed to measure regulatory relationships and therefore are not precise enough for this purpose. We provided examples to demonstrate this problem. Then we proposed a list of desirable characteristics of an association measure for gene regulatory relationships.
b. Following the desired characteristics identified in the previous step, we proposed a new association measure. The function is called *2D Visualized Co-regulation*. The function has been shown to perform better compared to the best known measures in the area. Its superiority is not only in the precision and recall of the function but also in its useful visualization characteristic. The visualization characteristic makes the result more convincing and plausible for the molecular biologist experts. It also has potential to give us additional information such as indirect relationships between the genes.
c. We also tried to show that our assumptions about the patterns of regulatory relationships in terms of machine learning were valid. These assumptions were used to create our 2D Visualized Co-regulation function. We applied a feature selection algorithm on our data which was then transformed into a grid. The grid was designed to present the relationship between two genes. It had the discretized value of each gene's expression along each axis and the content of each cell represents the frequency of that combination of gene values occurring at the same time in a sample. The feature selection algorithm detected the most important features inside the grid. The result of this experiment validated our assumptions.

   Similar to the experiment with the feature selection algorithm, we tried to prove the same thing in another way by using a K-means algorithm and a decision tree. To find the rules which can relate our class labels (ac, re, du) with the locations of

the densest area inside the grid. Both of these experiments supported our assumptions.

d. As a complementary process for the co-regulation function, we proposed a post-processing operation. We applied two post-processing operations to the output of our co-regulation function in order to prune the false positive records. The first idea again came from heuristics. We proposed looking for the absence of the opposite relationship for the pairs which had been labelled as having an activation or inhibition relationship. This means if the pair was recognized as having an activation relationship we looked for the absence of an inhibition pattern and vice a versa. When we applied this rule to the self regulatory relationships, we achieved a superior result. This post-processing step was demonstrated to be effective by reducing the number of false positives.

   The second post-processing method involved Data Processing Inequality (DPI) to reduce the false positives. DPI hypothesis is based on eliminating the weakest relationship among a loop of three genes. Based on our experiments, DPI also proved to be an effective process when it was applied to the result of our 2D Visualized Co-regulation function.

Specific Computational and Bioinformatics Contributions

Contribution to the Computer Science:

6. We explored and studied the functionality of the well known measures such as Pearson's correlation and Mutual Information in the context of GRN. The result of our study revealed that those functions do not measure the exact pattern of co-regulation and we needed to design a specific function for this purpose. We also found that there is not any literature related to the definition of regulatory relationship from the machine learning perspective.

Contributions to Bioinformatics:

7. Our first contribution to bioinformatics was introducing a new function for measuring co-regulation which was called *2D Visualized Co-regulation* function.

This function advanced the current performance of other pairwise association functions such as correlation measures. This function also provides us with a great visualization ability which has the potential to make the result sensible for biologists. The visualization feature of the function can also provide us with the additional information. Therefore, the function not only has a superior performance but also is more informative. This means two fold contributions.

8. The third contribution to the bioinformatics field was introducing a novel heuristic post-processing step. Our post-processing performance was proven to be quite effective when it was applied to the self regulatory pairs.
9. Our fourth contribution to the bioinformatics field was combining a further post-processing procedure (DPI) with our co-regulation function. DPI was previously used in the literature along with Mutual information. Here, we used it in combination with our co-regulation function and made a further improvement.

10.2.4 Approach 3: Heuristics Based on GRN Structure

The second major idea of approach 3 involved using structural properties of GRNs to construct the unknown GRN structure. The contributions in this part are listed:

a. The second major idea in the third approach was using a network of hub nodes to build the core structure of the target network. The procedure was started by detecting hub nodes from the known network to build a network of hubs by using microarray data. The degree of hub nodes was also extracted. We also identified additional hub nodes from microarray data, if any existed, and added them to that network. This gave us a core structure for the network. The overall procedure to build the network was incremental. In the second step, we built the second layer of the network by taking into account the degree of the hub nodes in the known network. We calculated the association of each hub node with any other gene and then we normalized these values and ranked them. Then we chose top listed genes from this ranked list according to the hub's degree and attached them to that hub. The hub's degree was then extracted from the domain knowledge. In the third step

we added the third layer to the network by adding nodes to the non hub nodes. For non hub nodes based on the literature again, we assumed a degree of one. The algorithm built the target network, layer by layer, taking into account all the above information. We also created a procedure which takes into account the skewness of the expression values in order to remove extreme values and noise.

b. Based on this study, it is highly recommended to use the hub's network in order to build the initial network structure. It is also quite helpful to use information about the degree distribution from the known networks.

Specific Computational and Biological Contributions

Our contribution to Bioinformatics:

10. We built a model which uses the *Hub Network* for the first time to build the core structure of an unknown GRN. This model employs several heuristic facts in addition to the Hub Network. The procedure for building the structure of the network is an incremental process. It takes into account the type of node (hub or non-hub), and the degree of the node to build the network. Information about the degree of the node was used to set up a background correction process where association measures for each gene normalized and then ranked and we chose the top ranked ones according to the degree of the node. This list was calculated for each gene and was used in an incremental process for adding nodes to the network. We also created a procedure based on the skewness of expression values in order to delete noise and extreme values.
11. This model can be used in conjunction with any correlation and association measure but we combined this model with our co-regulation function and achieved a superior performance compared to other well known systems in this area. This makes our last contribution in this approach.

In general, the third approach used the highest amount of heuristic information and achieved the best result among the approaches in this thesis. Our association measure has shown the ability to detect regulatory relationships more effectively compared with the other most used measures in this area. The combined performance of our 2D Visualized

Co-regulation function and Hub Network algorithm achieved a better prediction capacity than other well known studies.

In summary, in this thesis we investigated the effect of using heuristic information for GRN discovery and proved that using heuristics can improve the GRN discovery process. This was our last contribution to the bioinformatics field.

10.3 Limitation and Discussion

In this thesis, we introduced three new approaches for GRN discovery. In previous chapters we emphasised and discussed the impact of using each of these approaches in addition to their limitations. We mentioned that the first approach despite improving the state of the art performance of algorithms for solving differential equations was not scalable enough. The second approach did not perform well. The third approach was successful in improving the performance of the state of the art methods while being able to perform in real sized networks. Despite the success of the third approach, it had some limitations similar to any other method.

Our third approach uses heuristics from domain knowledge and has three components. The third component which works based on using known hubs relies on the availability of domain knowledge. In case that we do not have any information about the hubs, we will not be able to apply the first step of the algorithm. However the rest of the steps and the other two components of our system are still applicable and the performance is comparable with existing methods. One alternative would be to find the hubs from gene expression data by considering those genes which are associated with many other genes.

Another philosophical argument in the third approach is about the impact of using domain knowledge on the discovery of new information. Using information from domain knowledge is beneficial as it limits the search space when the search space is too large and the information content is low. In many previous applications, other researchers tried to embed heuristics from domain knowledge to improve the search process similar to what we did in the third approach. However, this might raise a question of the impact of using heuristics on the discovery of new information.

One should note that in this approach we do not build the connections based on domain knowledge but based on microarray gene expression data. In our third approach only the first layer uses information about known hubs to build the core structure of the network and the rest of the network is built based on the co-regulation measure of gene pairs and is independent from domain knowledge. Even in the first layer, we only use the list of known hubs, and the interaction between them is calculated based on the expression data. In case that we do not have any information about hubs, it is still possible to infer hubs from the gene expression data. The experiment in Chapter 9 shows that our algorithm is capable of finding new interactions as well as the existing interactions. The only exception is when there is a complex relationship rather than what we considered as ac, re and dual. In such a case our co-regulation function might not be able to correctly identify other complex relationships.

In this thesis we also used simulated data. It would be preferable to see the performance of the algorithms on some real datasets.

10.4 Directions for Future Work

The attempts in this thesis that we made to solve the problem of gene regulatory network discovery along with the comprehensive study of the literature gave us a good insight into the problem and the ability to identify possible future directions. Some of those ideas will now be discussed.

Approach 1: Using Memetic Gene Expression Programming

It will be possible to make a further improvement in Memetic Gene Expression technique by using a gradient search method instead of least mean square (LMS) for the local search.

Our Memetic Gene Expression Technique can be applied to any sort of regression and function finding problem in the future.

Approach 2: Using combined evolutionary algorithm

As mentioned in the second approach, we could not find a way to avoid the early convergence of our designed evolutionary algorithm therefore the local search proposal did not get a chance to be tested. One future direction is testing our proposed local search which uses cases from the domain knowledge to improve solutions. This proposal can be implemented in any suitable application where standard GA can be applied.

Approach 3: Using heuristics to build an association function

In this thesis we used a synthetic data generator, while the best result might be achieved with applying the third approach on a real dataset. Real benchmarks are essential for evaluation of algorithms in this area, but unfortunately there is a lack of them. The current ones such as (Werhli, Grzegorczyk et al. 2006; Cantone, Marucci et al. 2009) are limited in size and the network specification. For this reason we chose SynTReN simulator to test our algorithm.

Among other existing simulators, SynTReN produces networks which have similar structure with the known network, which is why we chose this particular one. Nevertheless, in terms of expression values it does not incorporate the domain knowledge. At the time of writing this thesis, a new GRN simulator was introduced called ReTRN which is claimed to produce not only a similar structure but also a similar expression profile (Li, Zhu et al. 2009). One possible future work is to further test our approaches by using ReTRN to produce synthetic data which resembles real data more closely.

The visualization effect of our 2D Visualized Co-regulation function that we claimed here needs to be tested properly in practice by involving biological experts.

There are also aspects that could be further considered in our third approach. One important aspect is considering dependency of more than two genes. Based on the facts from the literature that genes interact with each other in a small group of 4-5 (transcriptional groups); therefore it is necessary to consider dependency of more than two genes.

Another further consideration is improving the performance of our variable co-regulation function. In section 8.7, in the sixth experiment (Variable threshold Co-regulation Function), we showed that we can use machine learning to learn the pattern of regulatory relationships dynamically. This idea can be further investigated by using different encodings for transformation of the data inside the grid to the tabular format. The better encoding can preserve the existing pattern of the grid; therefore algorithms can learn the pattern better. A better discretization method for building the grid can also increase the amount of information provided and result in improving the performance. This was presented in Figure 8-10 in section 8.7. In addition, by showing more training data to the model, especially more "none" pairs to the model the performance can be increased further.

In this approach we involved only three types of regulatory relationships to define our co-regulation function. This idea can be extended to use the definition of more sophisticated gene relationships. There are more complex relationships that involve two or more genes including switches (Gardner, Cantor et al. 2000; Ham, Lee et al. 2008), oscillators (Stricker, Cookson et al. 2008; Purcell, Savery et al. 2010), cascading networks and logic gates which present more sophisticated control of gene expression. Implementation of each of those relationships as a function can be quite useful and result in increasing the predictability of GRNs (Mads, J.Blake et al. 2003).

In logic circuits theory and practice, some of these concepts are implemented as a separate module or chip to be used in the design of circuits. Similarly we can model these relationships and implement them as a function.

In approach 3: Using heuristics to build the Network Structure.

Our proposed Hub Network algorithm can be used in combination with other approaches such as Mutual Information or Bayesian network. Especially, this combination can be beneficial for Bayesian network modelling where finding the structure of the network is a challenging task. Our algorithm can assist a Bayesian Network approach to produce networks with more similar structure to the domain networks which is currently an issue.

For the future work, one could consider another recently observed structural property which we did not use in our modelling. The Bow-tie global architecture which is known to make the network's core (Csete and Doyle 2004; Kitano 2004; Ma'ayan 2009) can be used as well. This additional property might help algorithms to find the core structure of the network more effectively.

Another interesting idea is building a library of patterns of each gene containing the nature of the gene as a "fan in" or "fan out". This is especially important for hub genes. Usually hubs either have many outgoing links or many incoming links (Costanzo, Baryshnikova et al. 2010). This property is the signature of each gene and it is constant across networks in different conditions. Therefore, it can be used to build the core structure of the network more effectively and facilitate building a causal and directional network. Other information about each gene can be added also into such a library. For example the pattern of changes related to each gene like activation threshold and range of changes in its expression level (minimum and maximum values). This information can train a model which is going to measure the association between genes.

One further possible extension of the Hub Network is using the information about functional gene sets in combination with the Hub Network. Usually, hubs are important genes which are central genes for functional gene sets (Costanzo, Baryshnikova et al. 2010). We can retrieve the related gene sets to a hub and use information about other genes in the gene set to limit the search space. The functional gene sets were previously used to limit the search space in Module Network (Segal, Shapira et al. 2003) where a Bayesian Network approach was employed. However, they have not been used along with hubs in GRN discovery literature.

We discussed the possibility to use the PageRank algorithm (Page 1998) along with our Hub Network algorithm in Section 9.2. The PageRank algorithm takes into the account the importance of each node as well as the degree of that and has been used recently for gene network discovery in different ways (Özgür, Vu et al. 2008; Smriti, Vogel et al. 2009; Davis, Jr et al. 2010). It basically rank genes not only based on the number of connections but also based on the importance of those connections. Topic sensitive

PageRank algorithms (Haveliwala 2003) can also be used in this context. In the web pages application, topic sensitive PageRank algorithm considers the similarity of the query with web pages. Topics are Yahoo categories and algorithm biases the ranking procedure towards the more relevant pages related to the topic of the query. In other words, it captures more accurately the notion of importance with respect to a particular topic. Similar to this we can consider Gene Ontology categories or functional modules instead of Yahoo categories. Thus we can bias the ranking and selection procedure towards the genes which are related functionally in a process. Using Topic Sensitive PageRank we can bias the selection process towards genes which are functionally more related to each other or to the situation under study and in this way we can involve the experts' knowledge.

Future Direction of the GRN Discovery

Recent advances in biotechnology in the past years are shifting the technology of gene expression profiling from microarray to RNA sequencing. RNA sequencing results in a much more precise way of measuring genes expressions and promises a higher ability than microarray to discover Gene Regulatory Networks more effectively (Ozsolak and Milos 2011).

Compared to Microarray, RNA sequencing expression estimates are highly reproducible and often more accurate, less noisy and provide a larger dynamic range (Marioni, Mason et al. 2008). Microarray technology cannot fully catalogue and quantify gene expressions. The main reason is that the microarray probe sequence must be known a priori, where this can be done afterwards with sequence technology. Microarray gene expression profile measures continuous probe intensities and there are limited ways to analyse RNA splicing and other mechanisms affecting gene expression.

In addition to RNA sequencing, the availability of other genomics technologies can improve our knowledge of the interactions between different genomic areas and ultimately helps us build more precise GRNs.

Glossary

GRN: Gene Regulatory Network; a network which has genes as the nodes and regulatory relationship as edges.

Genome: Entire heredity information of an organism.

DNA: The most important molecules inside a cell containing codes for all function and operations of the cell.

Gene: Smallest functional part of a DNA which has a code to produce a protein.

Microarray: A glass or solid surface containing thousands of probes that have a gene attached to each probe. It can capture a profile of genes genome wide.

Association Measure: A measure (usually a mathematical formula) which specifies the strength of relationship or dependency between two variables.

Correlation Measure: Any statistical indicator of relationships between two or more random variables usually indicating a linear relationship.

Hub: Is a highly connected node in a network.

Small world Network: The small world property is often informally referred to as *six degrees of separation*. It is a type of network that most nodes can be reached from each other by a small number of steps. Many type of real networks such as social networks have this property.

Co-regulation: Genes that are expressed by the same transcriptional factor.

Co-expression: Genes that share similar expression patterns

Up regulation (Activation): Increase in the expression of a gene due to increase of expression of another gene or another factor.

Down regulation (Inhibition): Decrease in the expression of a gene due to increase of expression of another gene or another factor.

Dual Interaction: A relationship between two genes which sometimes is activation and sometimes is inhibition. lac_operon has a good example of such dual regulation (Müller-Hill 1996).

Gene Ontology: Is a big collection (database) which unifies representation of genes and gene products as well as providing a hierarchical classification of genes in groups based on their function or their product type.

Pathway (Biological pathway): Is a series of actions among molecules in a cell that leads to a certain product or a change in a cell.

Protein Interaction Network: An interaction network where nodes are cellular proteins and edges represent an interaction between them. Interactions can be physical (protein A binds protein B), metabolic (A and B catalyze reactions involving the same chemical), genetic (A and B are expressed together) or biochemical.

Bibliography

(2009). KEGG: Kyoto Encyclopedia of Genes and Genomes, Kanehisa Laboratories.

Aarts, E. H. L. and J. K. Lenstra (2003). Local Search in Combinatorial Optimization, Princeton Unviersity Press.

Abatangelo, L., R. Maglietta, et al. (2009). "Comparative study of gene set enrichment methods." BMC Bioinformatics 10(275).

Akitaya, T., A. Seno, et al. (2007). "Weak Interaction Induces an ON/OFF Switch, whereas Strong Interaction Causes Gradual Change: Folding Transition of a Long Duplex DNA Chain by Poly-l-lysine." Biomacromolecules 8(1): 273–278.

Al-Shahrour, F., R. Díaz-Uriarte, et al. (2005). "Discovering molecular functions significantly related to phenotypes by combining gene expression data and biological information." Bioinformatics 21(13).

Al-Shahrour, F., R. Díaz-Uriarte, et al. (2004). "FatiGO: a web tool for finding significant associations of Gene Ontology terms with groups of genes." Bioinformatics 20(4).

Al-Shahrour, F., P. Minguez, et al. (2006). "BABELOMICS: a systems biology perspective in the functional annotation of genome-scale experiments." Nucleic Acids Research 34(Web Server issue).

Albert, R. (2005). "Scale-free networks in cell biology." Journal of Cell Science 118(21): 4947-4957.

Allocco, D. J., I. S. Kohane, et al. (2004). "Quantifying the relationship between co-expression, co-regulation and gene function." BMC Bioinformatics 5(18).

Almaas, E., A. Vazquez, et al. (2007). Chapter 1: Scale-Free Networks In Biology. Biological Networks. F. Kepes, World Scientific Publishing.

Alon, U. (2007). "Network motifs: theory and experimental approaches." Nature Reviews Genetics 8: 450-461.

Altay, G. and F. Emmert-Streib (2010). "Revealing differences in gene network inference algorithm on the network-level by ensemble methods." Bioinformatics(May 25).

Arda, H. E. and A. J.M.Walhou (2009). "Gene-centered regulatory networks." Briefings in Functional Genomics **9**(14).

Areibi, S., M. Moussa, et al. (2001). A Comparison of Genetic/Memetic Algorithms and Other Heuristic Search Techniques. International Conference on Artificial Intelligence, , Las Vegas, Nevada.

Ashburner, M., C. A. Ball, et al. (2000). "Gene Ontology: tool for the unification of biology." Nature Genomics **25**: 25-29.

Banzhaf, W., Nordin, P., Keller, R.E., and Francone, F.D (1998). Genetic Programming: An Introduction: On the Automatic Evolution of Computer Programs and Its Applications, Morgan Kaufmann.

Barabási, A.-L., N. Gulbahce, et al. (2011). "Network medicine: a network-based approach to human disease." Nature Reviews Genetics **12**(1): 56-70.

Barabási, A.-L., M. E. J. Newman, et al. (2006). The structure and dynamics of networks, Princeton University Press.

Basso, K., A. A. Margolin, et al. (2005). "Reverse engineering of regulatory networks in human B cells." Nature Genetics **37**: 382–390.

Beer, D., S. L. Kardia, et al. (2002). "Gene-expression profiles predict survival of patients with lung adenocarcinoma." Nature Medicine **8**(8): 816-824.

Bhattacharjee, A., W. Richards, et al. (2001). "Classification of human lung carcinomas by mRNA expression profiling reveals distinct adenocarcinoma subclasses." PNAS **98**(24): 13790-13795.

Boslough, S. and P. A. Watters (2008). Statistics in a Nutshell, O'reilly.

Bramer, M. A. (2007). Principles of data mining. London, Springer.

Bulcke, T. V. d., K. V. Leemput, et al. (2006). "SynTReN: a generator of synthetic gene expression data for design and analysis of structure learning algorithms." BMC Bioinformatics **7**(43): 1471-2105.

Burke, E. K. and J. D. Landa-Silva (2004). The Design of Memetic Algorithms for Scheduling and Timetabling Problems. Recent Advances in Memetic Algorithms, Springer. **166/2005**.

Butte, A. J. and I. S. Kohane (1999). Unsupervised knowledge discovery in medical databases using relevance networks. AMIA Symp.

Butte, A. S. and I. S. Kohane (2000). Mutual information relvance networks: functional genomic clustering using pariwise entropy measurments. Pacific Symposium on Biocomputing.

Camillo, B. D., G. Toffolo, et al. (2009). "A Gene Network Simulator to Assess Reverse Engineering Algorithms." ANNALS of The New York Academy of Sciences **158**(The Challenges of Systems Biology Community Efforts to Harness Biological Complexity): 125-142.

Cantone, I. (2009). A Yeast Synthetic Network for In Vivo Assessment of Reverse-Engineering and Modeling Approaches P, European School of Molecular Medicine.

Cantone, I., L. Marucci, et al. (2009). "An yeast synthetic network for in vivo assessment of reverse engineering and modeling approaches." Cell **137**(1): 24-26.

Cantor, C. R. and C. L. Smith (1999). Genomics: the science and technology behind the Human Genome Project John Wiley & Sons.

Carr, R., W. Hart, et al. (2002). Alignment of Protein Structures with a Memetic Evolutionary Algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002), Morgan Kaufmann, New York, USA.

Cerulo, L., C. Elkan, et al. (2010). "Learning gene regulatory networks from only positive and unlabeled data." BMC Bioinformatics **11**(228).

Chen, T., V. Filkov, et al. (1999). Indentifying gene regulatory networks from experimental data. Proc. 3red Ann. Int. Conf. Comp. Mol. Biol. (RECOMB'99), New York, ACM Press.

Chickering, D. M., D. Heckerman, et al. (2004). "Large-Sample Learning of Bayesian Networks is NP-Hard." The Journal of Machine Learning Research **5**: 1287 - 1330.

Chuang, H.-Y., E. Lee, et al. (2007). "Network-based classification of breast cancer metastasis." Molecular systems Biology **3**(140).

Cleary, J. G. and L. E. Trigg (1995). An Instance-based Learner Using an Entropic Distance Measure. Proceedings of the12th International Conference on Machine Learning

Cline, M. S., M. Smoot, et al. (2007). "Integration of biological networks and gene expression data using Cytoscape." Nature Protocols **2**(10): 2366-2382.

Coleman, W. B. and G. J. Tsongalis (2002). The molecular basis of human cancer. New Jersey, HUMANA PRESS.

Corne, D. and C. Pridgeon (2004). "Investigating Issues in the Reconstructability of Genetic Regulatory Networks." Evolutionary Computation, 2004. CEC2004. **1**: 582 - 589.

Cornish-Bowden, A. (1995). Fundamentals of enzyme kinetics, Oxford University Press, USA.

Costanzo, M., A. Baryshnikova, et al. (2010). "The genetic landscape of a cell." Science **327**(5964): 425-431.

Cotta, C., A. Mendes, et al. (2003). Applying Memetic Algorithms to the Analysis of Microarray Data. Applications of Evolutionary Computing, Springer.

Cover, T. M. and J. A. Thomas (1991). Elements of Information Theory. New York, John Wiley & Sons.

Csete, M. and J. Doyle (2004). "Bow ties, metabolism and disease." Trends Biotechnology **22**(9): 446-450.

d'Alche-Buc, F. (2007). Inference of Biological Regulatory Networks: Machine Learning Approaches. Biological networks. F. Képès, World Scientific.

D'haeseleer, P. (2005). "How does gene expression clustering work?" Nature Biotechnology **23**(12): 1499-1502.

Dardel, F. and F. Kepes (2006). Bioinformatics: genomics and post-genomics, John Wiley and Sons.

Davierwala, A. P., J. Haynes, et al. (2005). "The Synthetic Genetic Interaction Spectrum of Essential Genes." Nature Genetics **37**(10): 1147-1152.

Davis, N. A., J. E. C. Jr, et al. (2010). "Surfing a genetic association interaction network to identify modulators of antibody response to smallpox vaccine." Nature Genes and Immunity **July**: 1-7.

Dawkins, R. (1976). The Selfish Gene. New York, Oxford University Press.

De Jong, H. (2002). "Modeling and simulation of genetic regulatory systems: a literature review." J Comput Biol **9**(1): 67–103.

Dempster, A. P. (1972). "Covariance Selection." Biometrics **28**(1): 157-175.

DeRisi, J. L., V. R. Iyer, et al. (1997). "Exploring the metabolic and genetic control of gene expression on a genomic scale." Science **24**(278): 680-686.

Dilão, R. and D. Muraro (2010). "A Software Tool to Model Genetic Regulatory Networks. Applications to the Modeling of Threshold Phenomena and of Spatial Patterning in Drosophila." PLoS ONE **5**(5).

Do, J. H. and D.-K. Choi (2007). "Clustering Approaches To Identifying Gene Expression Patterns from DNA Microarray Data." Mol. Cells **25**(2): 1-11.

Dubitzky, W., M. Granzow, et al. (2003). A Practical Approach to Microarray Data Analysis, Springer.

Eisen, M. B., P. T. Spellman, et al. (1998). "Cluster analysis and display of genome-wide expression patterns." PNAS **95**(25): 14863-14868

El-Mihoub, T. A., A. A. Hopgood, et al. (2006). "Hybrid Genetic Algorithms: A Review." Engineering Letters **13**(2).

Elowitz, M. B., A. J. Levine, et al. (2002). "Stochastic Gene Expression in a Single Cell." Science **297**: 1183-1186.

Emmert-Streib, F. and M. Dehmer (2009). Organizational Structure of the Transcriptional Regulatory Network of Yeast: Periodic Genes. Complex Sciences, Springer Berlin Heidelberg. **4:** 140-148.

Faith, J. J., B. Hayete, et al. (2007). "Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles." PLoS Biology **5**(1): e8.

Feist, A. M., M. J. Herrgård, et al. (2009). "Reconstruction of biochemical networks in microorganisms." Nature Reviews Microbiology **7**(2): 129-143.

Ferreira, C. (2001). "Gene Expression Programming: A New Adaptive Algorithm for Solving Problems." Complex Systems **13**(2): 87-129.

Ferreira, C. (2002). Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence, Springer.

Ferreira, C. (2008). What is Gene Expression Programming?

Filkov, V. (2006). Handbook of computational molecular biology, Chapman & Hall/CRC Taylor & Francis Group.

Fogel, D. B. and L. J. Fogel (1995). "An Introduction to Evolutionary Programming." Artificial Evolution **1063**: 21-33.

Fraley, C., A. E. Raftery, et al. (2006). mclust:Model-based Cluster Analysis. R package version 3.1-1.**:** http://www.stat.washington.edu/mclust.

Franke, L., H. v. Bakel, et al. (2006). "Reconstruction of a functional human gene network, with an application for prioritizing positional candidate genes." American journal of human genetics **78**(6): 1011-1025.

Fraser, K., Z. Wang, et al. (2010). Microarray image analysis: an algorithmic approach, CRC.

Friedman, N. and D. Koller (2003). "Being Bayesian About Network Structure. A Bayesian Approach to Structure Discovery in Bayesian Networks." Machine Learning **50**: 95-125.

Friedman, N., M. Linial, et al. (2000). "Using Bayesian Networks to Analyze Expression Data." JOURNAL OF COMPUTATIONAL BIOLOGY **7**(3/4): 601-620.

Fuente, A. d. l., N. Bing, et al. (2004). "Discovery of meaningful associations in genomic data using partial correlation coefficients." Bioinformatics **20**(18): 3565–3574.

Gama-Castro, S., V. Jimenez-Jacinto, et al. (2008). "RegulonDB (version 6.0): gene regulation model of Escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation." Nucleic Acids Research **36**(D120-4).

Gama-Castro, S., V. Jimenez Jacinto, et al. (2008). "gene regulation model of Escherichia coli K-12 beyond transcription, active (experimental) annotated promoters and Textpresso navigation." Nucleic Acids Research **36**.

Garber, M., O. Troyanskaya, et al. (2001). "Diversity of gene expression in adenocarcinoma of the lung." Proc Natl Acad Sci U S A. **98**(24): 13784-13789.

Gardner, T. S., C. R. Cantor, et al. (2000). "Construction of a genetic toggle switch in Escherichia coli." Nature **403**: 339-342.

Geberta, J., N. Raddea, et al. (2007). "Modeling gene regulatory networks with piecewise linear differential equations." European Journal of Operational Research **181**(3): 1148-1165.

Gehlenborg, N., S. I. O'Donoghue, et al. (2010). "Visualization of omics data for systems biology." Nature methods **7**(3): 56-68.

Gentleman, R. C., V. J. Carey, et al. (2004). "Bioconductor: Open software development for computational biology and bioinformatics." Genome Biology **5**: R80.

Gillespie, D. T. (1977). "Exact stochastic simulation of coupled chemical reactions." J. Phys. Chem **81**(25): 2340–2361.

Goeman, J., and Buhlmann P (2007). "Analyzing gene expression data in terms of gene sets: methodological issues." Bioinformatics **23**: 980-987.

Goemann B, Potapov AP, et al. (2009). "Comparative analysis of topological patterns in different mammalian networks." Genome Informatics **1**: 32-45.

Goh, K.-I., M. E. Cusick, et al. (2007). "The human disease network." PNAS **104**(21): 8685-8690.

Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Massachusetts, Addison-Wesley.

Goodman, L. A. and W. H. Kruskal (1954). "Measures of association for cross classifications." J. Amer. Statist. Assoc. **49**: 732-764.

Goodwin, B. C. (1963). Temporal organization in cells. New York, Academic Press.

Goodwin, B. C. (1965). "Oscillatory behavior in enzymatic control processes." Advances in enzyme regulation **3**: 425-428.

Gowda, T., S. Vrudhula, et al. (2009). "Prediction of Pairwise Gene Interaction Using Threshold Logic." Annals of the New York Academy of Sciences **1158**(The Challenges of Systems Biology Community Efforts to Harness Biological Complexity): 276-286.

Grzegorczyk, M., D. Husmeier, et al. (2008). Reverse Engineering Gene Regulatory Networks with Various Machine Learning Methods. Analysis of Microarray Data A Network-Based Approach. F. Emmert-Streib and M. Dehmer, WILEY-VCH Verlag GmbH& Co.kGaA.

Guyon, I. and A. Elisseeff (2003). "An Introduction to variable and feature selection." Journal of mahcine learning research **3**: 1157-1182.

Halinan, J. (2008). Chapter 4: Gene Network and Evolutionary Computation. Computational Intelligence in Bioinformatics G. Fogel, D. W. Corne and Y. Pan, John Wiley & Sons**:** 67-91.

Hall, M., E. Frank, et al. (2009). "The WEKA Data Mining Software: An Update." SIGKDD Explorations **11**(1).

Hall, M., E. Frank, et al. (2009). " The WEKA Data Mining Software: An Update; ." SIGKDD Explorations **11**(1).

Hallinan, J. (2008). Chapter4: Gene Networks and Evolutionary Computation. Computational Intelligence in BIOINFORMATICS. G. B. Fogel, D. W. Corne and Y. Pan, IEEE Press.

Hallinan, J. and A. Wipat (2006). Clustering and Cross-talk in a Yeast Functional Interaction Network. Computational Intelligence and Bioinformatics and Computational Biology.

Ham, T. S., S. K. Lee, et al. (2008). "Design and construction of a double inversion recombination switch for heritable sequential genetic memory." PLoS ONE **3**: e2815.

Hart, W. E., N. Krasnogor, et al. (2005). Recent Advances in Memetic algorithms, Springer Berlin / Heidelberg.

Haveliwala, T. H. (2003). "Topic-Sensitive PageRank: A Context-Sensitive Ranking Algorithm for Web Search." IEEE Transactions on Knowledge and Data Engineering **15**(4): 784-796.

Haynes, B. C. and M. R. Brent (2009). "Benchmarking regulatory network reconstruction with GRENDEL." Bioinformatics **25**(6): 801-807.

He, F., R. Balling, et al. (2009). "Reverse engineering and verification of gene networks: principles, assumptions, and limitations of present methods and future perspectives." Journal of biotechnology **144**(3): 190-203.

Heckera, M., S. Lambecka, et al. (2008). "Gene Regulatory Network Inference: Data Integration in dynamic models- A review." Bio Systems **96**(1): 86-103.

Hermsen, R., B. Ursem, et al. (2010). "Combinatorial Gene Regulation Using Auto-Regulation." PLoS Comput Biol **6**(6).

Hickman, G. J. and T. C. Hodgman (2010). "Inference of gene regulatory networks using boolean-network inference methods." Journal of bioinformatics and computational biology **7**(6): 1013-1029.

Hofmann, W.-K. (2006). Gene Expression Profiling by Microarrays: Clinical Implications, Cambridge University Press.

Hoops, S., S. Sahle, et al. (2006). "COPASI — a COmplex PAthway SImulator." Bioinformatics **22**: 3067-3074.

Imada, J. H. and B. J. Ross (2008). Using Feature-based Fitness Evaluation in Symbolic Regression with Added Noise. Genetic And Evolutionary Computation Conference (GECCO), Atlanta, GA, USA, ACM.

Jacob, F. and J. Monod (1961). "On the Regulation of Gene Activity." Cold Spring Harbor Symp. Quant. Biol **26**: 193-211.

Jiang, D., Z. Wu, et al. (2007). Parameter Identification in Differential equations by Gene Expression Programming. International Conference on Natural Computation (ICNC), IEEE.

Jiang, Z. and R. Gentleman (2007). "Extensions to gene set enrichment." Bioinformatics **23**(3): 306-313.

Junker, B. H. and F. Schreiber (2008). Analysis of biological networks, John Wiley & Son.

Kaern, M., W. J. Blake, et al. (2003). "The Engineering Of Gene Regulatory Networks." Annual Review of Biomedical Engineering **5**: 179-206.

Kauffman, S. (1969). "Homeostasis and differentiation in random genetic control networks." Nature **224**: 177-178.

Kaufman, S. A. (1979). Assessing the Probable Regulatory Structures and Dynamics of the Metazoan Genome. Kinetic logic : a Boolean approach to the analysis of complex regulatory systems. Berlin, Springer-Verlag. **29 of Lecture Notes in Bioinformatics:** 30-60.

Kenji Akiyama, Eisuke Chikayama, et al. (2008). "PRIMe: a Web site that assembles tools for metabolomics and transcriptomics." In Silico Biol **8**(0027).

Kepes, F. (2007). Biological Networks, World Scientific Publishing.

Kepes, F. (2007). Transcriptional Networks. Biological Networks. F. Kepes, World Scientific.

Keseler, I. M., C. Bonavides-Martinez, et al. (2009). "EcoCyc: A comprehensive view of Escherichia coli biology." Nucleic Acids Research **37**: D464-D470

Kikuchi, S., D. Tominaga, et al. (2003). "Dynamic modeling of genetic networks using genetic algorithm and S-system." Bioinformatics **19**(5).

Kimura, S., K. Ide, et al. (2004). "Inference of S-system models of genetic networks using a cooperative coevolutionary algorithm." Bioinformatics **21**(7): 1154-1163.

Kitano, H. (2004). "Biological robustness." Nature Reviews Genetics **5**: 826-837

Kitano, H. (2007). Scientific Challanges in Systems Biology. Introduction to systems biology. S. Choi, Humana Press Inc.

Kleinberg, J. (2000). The Small-World Phenomenon: An Algorithmic Perspective. Proceedings of the thirty-second annual ACM symposium on Theory of computing, ACM New York.

Kleinjan, D. and V. van Heyningen (2005). " Long-range control of gene expression: emerging mechanisms and disruption in disease. Am. J. Hum. Genet. 76:8-32." American Journal Humman Genetics **76**: 8-32.

Klipp, E., R. Herwig, et al. (2005). Systems biology in practice: concepts, implementation and application, WILEY-VCH Verlag GmbH & Co.

Konagurthu, A. S. and A. M. Lesk (2008). "On the origin of distribution patterns of motifs in biological networks." BMC Systems Biology **2**(73).

Koza, J. R. (1992). Genetic programming: on the programming of computers by means of natural selection. Cambridge, MIT Press.

Krasnogor, N., A. Aragón, et al. (2006). Chapter 12: MEMETIC ALGORITHMS. Metaheuristic Procedures for Training Neutral Networks, Springer US. **36**.

Krasnogor, N. and J. Smith (2005). "A Tutorial for Competent Memetic Algorithms: Model, Taxonomy and Design Issues." IEEE Transactions on Evolutionary Computation **A**(B): CCC 200D.

Lambert, J. (1991). Numerical methods for ordinary differential systems: the initial value problem, John Wiley & Sons.

Lee, H. K., W. Braynen, et al. (2005). "ErmineJ: Tool for functional analysis of gene expression data sets." BMC Bioinformatics **6**(269).

Lee, W.-P. and W.-S. Tzou (2009). "Computational Methods for Discovering Gene Networks from Expression Data." BRIEFINGS IN BIOINFORMATICS **10**(4): 408 - 423.

Lee, W. H., V. Narang, et al. (2009). "DREAM2 challenge." Annals of the New York Academy of Sciences **1158**: 196 - 204.

Leemput, K. V., T. V. d. Bulcke, et al. (2008). "Exploring the Operational Characteristics of Inference Algorithms for Transcriptional Networks by Means of Synthetic Data." Artificial Life **14**(1): 49-63.

Leonardson, A. S., J. Zhu, et al. (2010). "The effect of food intake on gene expression in human peripheral blood." Human Molecular Genetics **19**(1): 159-169.

Li X, Zhou C, et al. (2004). Investigation of Constant Creation Techniques in the Context of Gene Expression Programming. Evolutionary Computation Conference, GECCO2004.

Li, Y., Y. Zhu, et al. (2009). "ReTRN: A retriever of real transcriptional regulatory network and expression data for evaluating structure learning algorithm." Genomics **94**: 349-354.

Liang, S., S. Fuhrman, et al. (1998). "REVEAL, A GENERAL REVERSE ENGINEERING ALGORITHM FOR INFERENCE OF GENETIC NETWORK ARCHITECTURES." Pacific Symposium on Biocomputing **3**: 18-29.

Liu, G.-x., W. Feng, et al. (2009). "Reconstruction of Gene Regulatory Networks Based on Two-Stage Bayesian Network Structure Learning Algorithm." Journal of Bionic Engineering **6**: 86–92.

Lopes, H. S. and W. R. Weinert (2004). "EGIPSYS: an Enhanced Gene Expression Programming Approach for Symbolic Regression Problems." International Journal of Applied Mathematics and Computer Science **14**(3): 375--384.

Lu, X., V. V. Jain, et al. (2007). "Hubs in biological interaction networks exhibit low changes in expression in experimental asthma." Molecular systems Biology **3**(98).

Ma'ayan, A. (2009). "Insights into the Organization of Biochemical Regulatory Networks Using Graph Theory Analyses." Journal of Biological Chemistry **284**: 5451-5455.

Ma, H.-W., B. Kumar, et al. (2004). "An extended transcriptional regulatory network of Escherichia coli and analysis of its hierarchical structure and network motifs." Nucleic Acids Research **32**(22): 6643-6649.

Mabu, S., K. Hirasawa, et al. (2007). "A Graph-Based Evolutionary Algorithm: Genetic Network Programming(GNP) and its extension using Reinforcement Learning." Evolutionary Computation **15**(3): 369-398.

Madar, A., A. Greenfield, et al. (2010). "DREAM3: Network Inference Using Dynamic Context Likelihood of Relatedness and the Inferelator." PLoS ONE **5**(3).

Mads, K., W. J.Blake, et al. (2003). "The Engineering of Gene Regulatory Networks." Annual Review of Biomedical Engineering **5**: 179-206.

Maki, Y., D. Tominaga, et al. (2001). "Development of a system for the inference of large scale genetic networks." Pacific Symposium on Biocomputing **6**: 446-458.

Marbach, D., C. Mattiussi, et al. (2009). "Replaying the evolutionary tape: Biomimetic reverse engineering of gene networks." Annals of the New York Academy of Sciences **1158**: 234–245.

Marbach, D., R. J. Prillc, et al. (2010). "Revealing strengths and weaknesses of methods for gene network inference." PNAS **107**(14): 6286–6291.

Marghny, M. H. and I. E. El-Semman (2005). Extracting Logical Classication Rules With Gene Expression Programming: Microarray Case Study. AIML 05 Conference, Cairo, Egypt.

Margolin, A. A. and A. Califano (2007). "Theory and limitations of genetic network inference from microarray data." Ann N Y Acad Sci **1115**: 51-72.

Margolin, A. A., I. Nemenman, et al. (2006). "Aracne : An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context." BMC Bioinformatics.

Marioni, J. C., C. E. Mason, et al. (2008). "RNA-seq: An assessment of technical reproducibility and comparison with gene expression arrays." Genome Research **18**: 1509-1517.

Maslov, S. (2007). "Detecting topological patterns in protein networks." from http://www.cmth.bnl.gov/~maslov/3ieme_cycle_lecture_3_2007.ppt.

Maston, G., S. Evans, et al. (2006). "Transcriptional Regulatory Elements in the Human Genome." Annual Review of Genomics Human Genetics **7**: 29-59.

Mathivanan, S., B. Periaswamy, et al. (2006). "An evaluation of human protein-protein interaction data in the public domain." BMC Bioinformatics **7(Suppl 5)**(S19).

Matsumura, K., H. Oida, et al. (2005). "Inference of S-system models of genetic networks by function optimization using genetic programming." Transactions of the Information Processing Society of Japan **46**(11): 2814-2830.

Mendes, P., W. Sha, et al. (2003). "Artificial gene networks for objective comparison of analysis algorithms." Bioinformatics **19**(2): ii122–ii129.

Menezes, M. D. and A. Barabási (2004). "Fluctuations in network dynamics." Physical review letters **92**(2).

Merz, P. (2000). Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscapes and Effective Search Strategies PhD, University of Gesamthochschule Siegen.

Merz, P. and B. Freisleben (1999). Fitness Landscapes and Memetic Algorithm Design, McGraw--Hill, 1999.

Merz, P. and A. Zell (2002). Clustering Gene Expression Profiles with Memetic Algorithms based on minimum spanning trees. Parallel Problem Solving from Nature - PPSN VII: 7th International Conference, Granada, Spain, Springer.

Meyer, P. E., K. Kontos, et al. (2007). "Information-theoretic inference of large transcriptional regulatory networks." EURASIP J Bioinform Syst Biol(79879).

Meyer, P. E., F. Lafitte, et al. (2008). "minet: A R/Bioconductor Package for Inferring Large Transcriptional Networks Using Mutual Information." BMC Bioinformatics **9**(461): 1471-2105.

Milo, R., S. Shen-Orr, et al. (2002). "Network motifs: simple building blocks of complex networks." Science **298**(5594): 824 - 827.

Mootha, V. K., C. M. Lindgren, et al. (2003). "PGC-1-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes." Nature **34**: 267 - 273.

Mootha, V. K., C. M. Lindgren, et al. (2003). "PGC-1-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes." Nature Genetics **34**: 267 - 273.

Morishita, R., H. Imade, et al. (2003). "Finding Multiple Solutions Based on An Evolutionary Algorithm for Inference of Genetic Networks by S-system." Shisutemu, Joho Bumon Gakujutsu Koenkai Koen Ronbunshu: 155-160.

Moscato, P. (1989). "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms."

Moscato, P. and M. Norman (1989). A competitive-Cooperative Approach to Complex Combinatorial Search, Caltech Concurrent Computation Program.

Moscato, P. and M. Norman (1992). A" Memetic" Approach for the Traveling Salesman Problem Implementation of a Computational Ecology for Combinatorial Optimization on Message-Passing Systems. International Conference on Parallel Computing and Transputer Applications, Amsterdam, IOS Press.

Müller-Hill, B. (1996). The lac Operon: a short history of a genetic paradigm. Berlin, Walter de Gruyter & Co.

Muller, H., E. Kenny, et al. (2004). "Textpresso: an ontology-based information retrieval and extraction system for biological literature." PLoS Biology **2**(11).

Müssel, C., M. Hopfensitz, et al. (2010). "BoolNet--an R package for generation, reconstruction and analysis of Boolean networks." Bioinformatics **26**(10): 1378-1380.

Nachman, I., A. Regev, et al. (2004). "Inferring Quantitative Models of Regulatory Networks from Expression Data." Bioinformatics **20** (suppl 1): i248-i256.

Nicolis, G. and I. Prigogine (1977). Self-organization in nonequilibrium systems, Wiley New York.

Noman, N. and H. Iba (2005). Inference of gene regulatory networks using s-system and differential evolution. Proceedings of the 2005 conference on Genetic and evolutionary computation.

Noto, K. and M. Craven (2005). Learning Regulatory Network Models that Represent Regulator States and Roles. RECOMB 2004 Ws on Regulatory Genomics, Springer-Verlag Berlin.

Novick, A. and M. Weiner (1957). "Enzyme induction as an all-or-none phenomenon." Proc. Natl. Acad. Sci. USA **43**: 553-566.

Özgür, A., T. Vu, et al. (2008). "Identifying gene-disease associations using centrality on a literature mined gene-interaction network." Bioinformatics **24**(13): 277-285.

Ozsolak, F. and P. M. Milos (2011). "RNA sequencing: advances, challenges and opportunities." Nature Review Genetics **12**(February): 87-98.

Page, L. (1998). Method for node ranking in a linked database. U.S.A, The Board of Trustees of the Leland Stanford Junior University.

Parmigiani, G., E. S. Garrett, et al. (2003). The analysis of gene expression data: methods and software, Springer London.

Pei, P. and A. Zhang (2005). A Topological Measurement for Weighted Protein Interaction Network. Computational Systems Bioinformatics Conference, 2005. Proceedings. 2005 IEEE.

Potapov, a. P. (2008). SIGNAL TRANSDUCTION AND GENE REGULATION NETWORKS. Analysis of Biological Networks. B. H. Junker, Schreiber, Falk. New Jersey, John Wiley & Sons,.

Prill, R. J., P. A. Iglesias, et al. (2005). "Dynamic Properties of Network Motifs Contribute to Biological Network Organization." PLoS Biology **3**(11): e343.

Purcell, O., N. Savery, et al. (2010). "A comparative analysis of synthetic genetic oscillators." Journal of the Royal Society Interface **June**.

Purnick, P. E. M. and R. Weiss (2009). "The second wave of synthetic biology: from modules to systems." Nature **10**: 410-422.

Qiu, P., A. J. Gentlesa, et al. (2009). "Fast calculation of pairwise mutual information for gene regulatory network reconstruction." Computer Methods and Programs in Biomedicine **94**(2).

Ramsey, C. L. and J. J. Grefenstette (1993). Case-based initialization of genetic algorithms. 5th International Conference on Genetic Algorithms, Morgan Kaufmann Publishers Inc.

Ravasz, E., A. L. Somera, et al. (2002). "Hierarchical Organization of Modularity in Metabolic Networks." Science **297**(5586): 1551 - 1555.

Rechenberg, I. (1971). Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution PhD, Technical University Berlin.

Reeves, C. R. (1993). Using genetic algorithm with small populations. Fifth International Conference on Genetic Algorithms.

Reeves, C. R. (2003). Genetic Algorithms. HandBook of Metaheuristics. F. Glover and G. A. Kochenberger, Kluwer Academic Publishers.

Reinitz, J. and J. R. Vaisnys (1990). "Theoretical and experimental analysis of the phage lambda genetic switch implies missing levels of co-operativity." Journal of Theoretical Biology **145**(3).

Riolo, R., T. Soule, et al. (2008). Genetic programming theory and practice V, Springer.

Rogers, D. (1991). "A Hybrid of Friedman's multivariate adaptive regression splines(MARS) algorithm with Holland's genetic algorithm." Proceedings of the Fourth International conference on Genetic Algorithms: 384-391.

Roth, F. (2005). Introduction to Protein Interaction Network Analysis, Harvard Medical School,MGH Institute for Neurodegenerative Disease.

Roy, S., M. Werner-Washburne, et al. (2008). "A system for generating transcription regulatory networks with combinatorial control of transcription." Bioinformatics **24**(10): 1318-1320.

Sakamoto, E. and H. Iba (2001). Inferring a System of Differential Equations for a Gene Regulatory Network by using Genetic Programming. Proceedings of the 2001 IEEE congress on Evolutionary Computation.

Sauro, H. M., D. Harel, et al. (2006). Challenges for Modeling and Simulation Methods in Systems Biology. Proceedings of the 38th conference on Winter simulation, Winter Simulation Conference

Schadt, E. E., K. Stefansson, et al. (2008). "Genetics of Human Gene Expression and Gene-Gene Transcriptional Networks." Nature Genomics.

Schäfer, J. and K. Strimmer (2005). "A shrinkage approach to large-scale covariance matrix estimation and implication for functional genomics." Statistical applications in genetics and molecular biology **4**(32).

Scherer, A. (2009). Batch Effects and Noise in Microarray Experiments: Sources and Solutions By Andreas Scherer, John Wiley & Sons.

Schlitt, T. and A. Brazma (2007). "Current Approaches to Gene Regulatory Network Modelling." BMC Bioinformatics **8**(Suppl 6).

Schwefel, H.-P. (1974). Numerische Optimierung von Computer-Modellen PhD, Technical University of Berlin.

Segal, E., M. Shapira, et al. (2003). "Module networks: Identifying regulatory modules and their condition-specific regulators from gene expression data." Nature Genetics **34**(2): 166 - 176.

Sehgal, M. S. B., I. Gondal, et al. (2007). Computational Intelligence Methods for Gene Regulatory Network Reconstruction. Computational Intelligence in Medical Informatics, Springer-Verlag.

Seidel, C. (2008). Analysis of Microarray Data A Network-Based Approach, WILEY-VCH Verlag GmbH.

Seo, C. H., J.-R. Kim, et al. (2009). "Hub Genes With Positive Feedbacks Function as Master Switches in Developmental Gene Regulatory Networks." Bioinformatics **25**(15): 1898-1904.

Sestito, S. and T. Dillon (1991). "Using single-layered neural networks for the extraction of conjunctive rules and hierarchical classifications." Applied Intelligence **1**(2): 157-173.

Sestito, S. and T. Dillon (1993). "Knowledge acquisition of conjunctive rules using multilayered neural networks." International Journal of Intelligent Systems **8**(7): 779-805.

Shannon, P., A. Markiel, et al. (2003). "Cytoscape: A Software Environment for Integrated Models of Biomolecular Interaction Networks." Genome Research **13**: 2498-2504.

Sinha, A. and D. E. Goldberg (2003). A Survey of Hybrid Genetic and Evolutionary Algorithms, Illinois Genetic Algorithms Laboratory, University of Illinois.

Sjöberg, P., P. Lötstedt, et al. (2007). "Fokker–Planck approximation of the master equation in molecular biology." Computing and Visualization in Science **12**(1).

Smolen, P., D. A. Baxter, et al. (2006). Modeling transcriptional control in gene networks—methods, recent results, and future directions, Springer New York.

Smriti, R. R., C. Vogel, et al. (2009). "Mining gene functional networks to improve mass-spectrometry-based protein identification." Bioinformatics **25**(22): 2955-2961.

Smyth, G. K. (2004). "Linear models and empirical Bayes methods for assessing differential expression in microarray experiments." Statistical Applications in Genetics and Molecular Biology **No. 1, Article 3**.

Smyth, G. K. (2005). limma: Linear Models for Microarray Data, Springer, New York.

Smyth, G. K. (2005). Limma: linear models for microarray data. In: Bioinformatics and Computational Biology Solutions using R and Bioconductor. Bioinformatics and Computational Biology Solutions using R and Bioconductor, Springer, New York.

Soranzo, N., G. Bianconi, et al. (2007). "Comparing association network algorithms for reverse engineering of large-scale gene regulatory networks:

synthetic versus real data." Bioinformatics **23**(13): 1640–1647.

Speer, N., P. Merz, et al. (2003). "Clustering Gene Expression Data with Memetic algorithms based on Minimum Spanning Trees." Evolutionary Computation **3**: 1848 - 1855.

Spieth, C., F. Streichertr, et al. (2004). Optimizing Topology and Parameters of Gene Regulatory Network Models from Time-Series Experiments. GECCO 2004, Springer-Verlag Berlin Heidelberg.

Stricker, J., S. Cookson, et al. (2008). "A fast, robust and tunable synthetic gene oscillator." Nature **456**: 516-519.

Stuetzle, T. G. (1998). Local Search Algorithms for Combinatorial Problems PhD, Universit¨at Darmstadt.

Subramanian, A., P. Tamayo, et al. (2005). "Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles." PNAS,Proceedings of the National Academy of Sciences **102**(43).

Sun, J., K. Tuncay, et al. (2007). "Transcriptional regulatory network discovery via multiple method integration: application to e. coli K12." Algorithms Molecular Biology **2**(2).

Swain, P. S., M. B. Elowitz, et al. (2002). "Intrinsic and extrinsic contributions to stochasticity in gene expression." PNAS **99**(20).

Tan, P.-N., M. Steinbach, et al. (2006). Introduction to data mining, Pearson Education Inc.

Tanay, A., R. Sharan, et al. (2004). "Revealing modularity and organization in the yeast molecular network by integrated analysis of highly heterogeneous genomewide data." PNAS **101**(9): 2981-2986.

Theocharidis, A., S. v. Dongen, et al. (2009). "Network visualization and analysis of gene expression data using BioLayout Express." Nature Protocols **4**: 1535 - 1550.

Thieffry, D., A. M. Huerta, et al. (1998). "From specific gene regulation to genomic networks: a global analysis of transcriptional regulation in Escherichia coli." BioEssays **20**(5): 433–440.

Thomas, R. (1981). On the relation between the logical structure of systems and their ability to generate multiple steady states or sustained oscillations. Numerical Methods in the Study of Critical Phenomena, Springer-Verlag, Berlin.

Thomas, R., D. Thieffry, et al. (1995). "Dynamical behaviour of biological regulatory networks—I. Biological role of feedback loops and practical use of the concept of the loop-characteristic state." Bulletin of Mathematical Biology **57**(2).

Van den Bulcke, T. (2009). Personal Communication.

van Rijsbergen, C. V. (1979). Information Retrieval. London; Boston. Butterworth.

Vázqueza, A., A. Flamminia, et al. (2003). "Modeling of Protein Interaction Networks." Complexus **1**: 38-44.

Veer, L. J., H. Dai, et al. (2002). "Gene expression profiling predicts clinical outcome of breast cancer." Nature **415**: 530-536.

Vilcota, G. and J.-C. Billaut (2008). "A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problem." European Journal of Operational Research **190**(2): 398-411.

Wang, H., L. Qian, et al. (2010). "Inference of gene regulatory networks using S-system: a unified approach." IET Syst Biol **4**(2): 145-156.

Wang, Y., T. Joshi, et al. (2006). "Inferring gene regulatory networks from multiple microarray datasets." Bioinformatics **22**(19): 2413-2420.

Watkinson, J., K. C. Liang, et al. (2009). "Inference of regulatory gene interactions from expression data using three-way mutual information." Ann N Y Acad Sci **1158**: 302-313.

Weaver, D. C., C.T.Workman, et al. (1999). Modeling Regulatory Networks with Weight Matrices. Pacific Symposium on Bioinfomratics.

Werhli, A. V., M. Grzegorczyk, et al. (2006). "Comparative Evaluation of Reverse Engineering Gene Regulatory Networks with Relevance Networks, Graphical Gaussian Models and Bayesian Networks." Bioinformatics **22**(20): 2523-2531.

Wilson, S. R. and Y. E. Pittelkow (2007). From Gene Expression to Clinical Diagnostic Tool- Are we there yet? Winter School in computational biology, Brisbane.

Wilson, S. W. (2008). Classifier Conditions Using Gene Expression Programming, University of Illinois at Urbana-Champaign, USA.

Wolpert, D. H., Macready, W.G. (1997). "No Free Lunch Theorems for Optimization." IEEE Transactions on Evolutionary Computation **1**(67).

Wu, D., F. o. Vaillant, et al. (2010). "ROAST: rotation gene set tests for complex microarray experiments." Bioinformatics.

Xiao, Y. (2009). "A Tutorial on Analysis and Simulation of Boolean Gene Regulatory Network Models." Current Genomics **10**: 511-525.

Yang, X., J. L. Deignan, et al. (2009). "Validation of Candidate Causal Genes for Abdominal Obesity Which Affect Shared Metabolic Pathways and Networks." Nature Genetics **41**(4): 415-423.

Yang, Z., K. Tang, et al. (2008). "Large scale evolutionary optimization using cooperative coevolution." Information Sciences **178**(15): 2985-2999.

Yeung, K. Y., M. Medvedovic, et al. (2004). "From co-expression to co-regulation: how many microarray experiments do we need?" Genome Biology **5**(7).

Yu, J., V. A. Smith, et al. (2004). "Advances to Bayesian Network Inference for Generating Causal Networks from Observational Biological Data " Bioinformatics **20**(18): 3594-3603.

Yu, X. and M. Gen (2010). Introduction to Evolutionary Algorithms. New York, Springer Heidelberg.

Zak, D. E., F. J. Doyle, et al. (2001). Simulation studies for the identification of genetic networks from cDNA array and regulatory activity data. Proc. 2nd Intl. Conf. Systems Biology.

Zhang, B. and S. Horvath (2005). "A general framework for weighted gene co-expression network analysis." Statistical applications in genetics and molecular biology **4**(17).

Zhao, J., H. Yu, et al. (2006). "Hierarchical modularity of nested bow-ties in metabolic networks." BMC Bioinformatics **7**(386).

Zhou, X. J., J. M.-c. Kao, et al. (2005). "Functional Annotation and Network Reconstruction through Cross-platform Integration of Microarray Data." Nature Biotechnology **23**(2): 238-243.

Zhu, J., B. Zhang, et al. (2008). "Integrating Large-Scale Functional Genomic Data to Dissect the Complexity of Yeast Regulatory Networks." Nature Genomics.