Media Engineering and Technology Faculty
German University in Cairo

Identifying Key Regulators in Gene Regulatory Networks

Bachelor Thesis

Author:	Amira A. M. Gamaleldin Zaki
Supervisors:	Prof. Dr. Rolf Backofen [1]
	Dr. Sebastian Will [1]
	Martin Mann [1]
Reviewer:	Prof. Dr. Slim Abdennadher [2]
Submission Date:	30 August, 2007

[1] Albert Ludwigs University, Freiburg, Germany

[2] German University in Cairo, Egypt

This is to certify that:

(i) the thesis comprimises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

Amira A. M. Gamaleldin Zaki
30 August, 2007

Acknowledgement

I would like to express my gratitude to Prof. Dr. Rolf Backofen for inviting me to the Bioinformatics department at Albert Ludwigs University. His warm welcome, motivational support and academic guidance has been of great help throughout my stay in Freiburg.

I am deeply indebted to my supervisor, Martin Mann, for his constant support throughout my stay. His brilliant ideas and suggestions have inspired my work and his comments and thoughts have been of great use for writing my thesis.

In addition, I would like to thank Dr. Sebastian Will for his ideas and for enriching my knowledge in the Bioinformatics field. Moreover, I would like to thank everyone at the Bioinformatics department for their hospitality and friendliness.

I am also indebted to Prof. Dr. Slim Abdennadher for encouraging me to research in the field of Bioinformatics and giving me the possibility to join Prof. Backofen's group. In addition, I cannot forget to thank the authorities at the German University in Cairo and the German Academic Exchange Service (DAAD) for granting me the financial support to perform my thesis in Germany.

Abstract

Bioinformatics is an emerging discipline that uses information technology to organize, analyze, and distribute biological information in order to answer complex biological questions and uncover new biological insights. The complex assemblages of interconnected genes, proteins and other molecules are represented as gene regulatory networks. Within these networks are key regulators that are responsible for many essential cell processes and have various effects on the characteristics of an organism. This work presents a model for gene regulatory networks and implements a means to identify optimal key regulators within the network. In addition, cycles representing positive or negative feedback loops increase the complexity of the task; a means to integrate the effect of such cycles into the model has been implemented.

Contents

Chapter 1

Introduction

Huge advances in the fields of molecular biology and genomics have greatly increased the amount of biological information available. This explosion in the amount of data available has necessitated the need to introduce computer technologies to manage and analyze the data store. Thus the field of bioinformatics has arisen in order to uncover the wealth of biological information hidden in the mass of data and obtain a clearer insight into the fundamental biology of organisms. This new knowledge could have profound impacts on fields as varied as human health, agriculture, the environment, energy and biotechnology.

In particular, genome sequencing projects have made huge amounts of data available. Analysis of these data has contributed to the discovery of a large number of genes and their regulatory sites. The KEGG database, for example, currently contains information on the structure and function of about 110,000 genes for 29 species [10]. In some cases, the proteins involved in the regulation of the expression of these genes have been identified, as well as the molecular mechanisms through which they achieve this. The complex patterns of behaviour from the interactions between genes, proteins and other molecules are known as gene regulatory networks. Gaining an understanding of these networks creates an understanding of the multiple interactions in a cell and can be a huge scientific challenge with potentially high industrial benefits.

Within a network, exist certain regulatory entities which are capable of having control over other genes. Amongst them are key regulators which are sufficient to induce an entire complex developmental pathway within an organism. Identification of these key regulatory components has provided useful insight into the developmental process and has enormous potential for controlling them. Thus, the development of approaches to identify key regulators and understand where they reside in regulatory hierarchies is expected to be extremely valuable.

As most genetic regulatory networks of interest involve many genes connected through interlocking feedback loops, an intuitive understanding of their behaviour is hard to obtain. By using computational simulations and mathematical representations to model them, the behaviour of possibly large and complex regulatory systems can be predicted and explained in a systematic way.

Since the 1960s, a variety of mathematical formalisms to describe regulatory networks have been proposed. These formalisms are complemented by simulation techniques to make behavioural predictions from the model of a system, as well as modelling techniques to construct the model from experimental data and knowledge on regulatory mechanisms. Formalisms can be directed graphs, Bayesian networks, Boolean networks, ordinary and partial differential equations and stochastic equations [7].

The level of detail determines the modelling approach and the formalism used. A logical or binary approach is one where each gene is treated as on or off, and the dynamics describe how groups of genes act to change one another's states over time. This is a simple model that can be represented using directed graphs, Bayesian networks or Boolean networks. However, it is difficult to include the many details of cellular biology in them. A more detailed description is the chemical kinetics or rate-equation approach in which the variables of interest are the concentrations of individual proteins within the cell and the dynamics describe the rates of production and decay of these proteins. This more complex model uses ordinary and partial differential equations. An even more detailed approach is the stochastic kinetics one, using stochastic equations. It models the techniques for simulating the behaviour of chemical reactions, involving small number of molecules, by applying them to the reactions involved in protein-DNA interactions. This approach is quite complete, yet it has a high computational cost and scarifies any immediate prospect of analytical treatment [9].

In this work, gene regulatory networks will be modelled as directed graphs using the logical approach. It reduces the interactions between genes into simple terms, which can be used to gain further understanding into the global interactions between entities cooperating in complex cellular processes. Then the aim is to produce an algorithm to identify key regulator components within the networks.

This thesis is organized as follows: This chapter is an introduction of the bioinformatics research field and the chosen modelling problem. The next chapter presents an overview of the biological background needed to fully comprehend the basis of the problem. In chapter 3, the problem is stated and then the general algorithm suggested to solve it. After that is a discussion of the different models attempted to solve the problem. Following is chapter 5 which contains a discussion of the implementation performed to test the models and algorithms. The next chapter is a discussion of the results obtained by testing the implementation on real biological data. The last chapter contains a conclusion, summarizing the work performed and discussing the open topics for future research.

Chapter 2

Biological background

Every organism has a genome that encodes the biological information needed for its existence, development and reproduction. Organisms are either made up of cells and known as cellular organisms, or they are made up of the genetic information surrounded by a protein coat and known as non-cellular organisms. The genomes of all cellular organisms, including humans, are made up of *DNA (Deoxyribonucleic Acid)*. Genomes of many non-cellular viruses are made up of *RNA (Ribonucleic Acid)*. Within each organism, these two types of genomes are interchangeable.

DNA is a long polymer made up of two strands of *nucleotides* connected to one another by hydrogen bonds. The molecule strands wind together through space to form a structure described as a double helix. Each nucleotide is a molecule made up of a sugar, a phosphate and a base. In DNA there are four types of bases: adenine (A), thymine (T), cytosine (C), and guanine (G). The sequence of these four bases is called the *DNA sequence*; this sequence encodes the genetic information required to produce a particular organism and maintain its own unique traits. Thus the main role of DNA in the cell is the long-term storage of information; since it contains the instructions to construct other components of the cell, such as proteins and RNA molecules. The DNA segments that carry this genetic information are called *genes*, but other DNA sequences have structural purposes, or are involved in regulating the expression of genetic information.

RNA is a nucleic acid consisting of nucleotide monomers (which are simple molecules). It is single-stranded and consists of the same four types of bases as DNA, but the thymine base is replaced with uracil (U). It is synthesized from DNA by enzymes called RNA polymerases and further processed by other enzymes. RNA directs the synthesis of proteins as it serves as the template for translation of genes into proteins [1, 4].

The process by which cells produce proteins from the gene instructions encoded in the DNA is known as the *Gene expression*. First the nucleotide sequence of the appropriate portion of the long DNA molecule (i.e. gene) is copied into RNA; this process is called *transcription*. Then the RNA copies are used directly as templates to direct the synthesis of the protein, this process is called *translation*. The flow of genetic information in cells is therefore from DNA to RNA to protein as in figure 2.1. At each step regulatory molecules, known as *transcription factors*, control the concentration and form of product by repressing or activating the reactants [8]. Since these transcription factors are themselves products of genes, the ultimate effect is genes regulating each other's expression by forming so called *gene regulatory networks*.

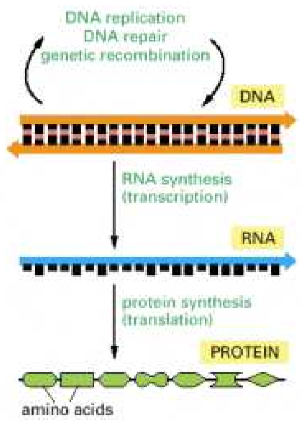

Figure 2.1: The flow of genetic information from DNA to RNA (transcription) and from RNA to protein (translation) occurs in all living cells [16].

Definition 2.0.1 *Gene regulatory networks* are a method of representing the complex assemblages of interconnected genes, proteins and other molecules. They are used to study the concerted action of genes during cell differentiation and other essential processes. They are depicted as nodes connected by edges. The *nodes* correspond to the genes, messengers or proteins and the *edges* represent regulatory interactions (activations or repressions) between the nodes [2].

The cell types in a multicellular organism become different from one another because they synthesize and accumulate different sets of RNA and protein molecules. They generally do this without altering the sequence of their DNA, but rather by expressing different sets of genes. Thus the basis of cell differentiation is differential gene expression, and this is accomplished by interactions between genes, i.e. gene regulatory networks. Genes that are expressed are known to be *activated* and those which are not expressed are known to be *repressed*. Thus the interactions in gene regulatory networks denote activations and repressions of genes.

A living cell contains thousands of enzymes, many of which operate at the same time. A *metabolic pathway* is a name given to the series of chemical reactions occurring within a cell catalyzed by enzymes. The cell controls how many molecules of each enzyme it makes by regulating the expression of the gene that encodes that enzyme. Thus the complex webs of metabolic pathways are examples of gene regulatory networks.

Moreover, cells can change the pattern of genes they express in response to changes in their environment, such as signals from other cells. For example, if a liver cell is exposed to a hormone called glucocorticoid, then the production of several specific proteins is greatly increased. During periods of starvation or intense exercise, this hormone is released in the body and signals the liver to increase the production of enzymes that produce glucose from amino acids and other small molecules. When the hormone is no longer present, the production of the enzymes drops to its normal level [1]. Such interactions leading to change in the gene expression is indicated as a *signal transduction network* which is a gene regulatory network.

Within these networks, *regulators* are those factors having control over other genes. They are capable of activating or repressing the others, hence regulating the expression of the gene and thus the function and characteristics of the cell. Furthermore, *key (or master or global) regulators* are a type of regulators having control in several different metabolic pathways; thereby having multiple effects on the phenotype (characteristics) of an organism [11]. For example, in Drosophila melanogaster (a type of insect) the expression of the transcription factor eyeless induces unusual eye formation, while in vertebrates, expression of the proteins MyoD, Myf5, and NeuroD can induce muscle and neural development, respectively [3]. It follows that it would be useful to be able to identify these key regulatory components, since they provide a useful insight into developmental processes and have an enormous potential for controlling them. Therefore the development of approaches to identify key regulators and understand where they reside in regulatory hierarchies is expected to be extremely valuable.

Auto-regulation is a common feature available in gene regulatory networks. It is a property of a network whereby a component of the system controls its own activity [9]. If a component inhibits its own level of activity then it is in a *negative feedback cycle*. An example of this is when an enzyme acting early in a reaction pathway is inhibited by a late product of that pathway. Thus, whenever large quantities of the final product begin to accumulate, this product binds to the first enzyme and slows down its catalytic action, thereby limiting the further entry of reactants into that reaction pathway [1]. Activating a negative feedback cycle increases the stability in gene regulatory systems since it helps in bringing activation levels back to normal, whereas inhibiting it has no biological meaning.

On the other hand, if a component increases its own level of activity then it is in a *positive feedback cycle*. For example, an enzyme can generate a product that binds back to the enzyme, further increasing the enzyme's activity. Such cycles decrease stability in a network since they enhance oscillations in the behaviour of the network. Thereby, activating a positive feedback cycle causes an increase in activation levels, whereas inhibiting it has no biological meaning [9].

Chapter 3

Methods

3.1 Problem Statement

As discussed in the previous chapter, gene regulatory networks are methods of representing the complex assemblages of interconnected genes in an attempt to study their interactions. In a binary approach, where genes are considered as entities capable of being turned on or off, it is required to *suggest different models for modelling genetic regulatory networks.* First, the model considered is an abstract one that simply shows the genes and their connections. Then a more complex model is produced that shows the type of interactions and the complexity of propagating such interactions. In the end, it is modified into a model that handles negative and positive feedback loops.

All the models are based on the modelling of a gene regulatory network as a *directed graph*, as defined below. The definitions are extended according to the specifications of each model.

Definition 3.1.1 A *directed graph* is defined as a *tuple* $< N, E >$*;* where

N is the set of nodes and E is the set of edges. Each *node* n has a set R_n, which is a set consisting of the nodes that it regulates. Formally this means $n \in N \rightarrow R_n \subseteq N$. Each *edge* in E is defined as a tuple $< i, j >$, where $i, j \in N$, i represents the head of the edge and j represents the tail of the edge.

A *path* from a node n to a node n' is a sequence of vertices starting with n and ending with n' connected by graph edges. A *cycle* is a path starting and ending at the same point. A *cyclic* graph is a graph that contains cycles. An *acyclic graph* is a cycle free graph containing no cycles; thus for any path in the graph the end and start vertices are distinct [5]. Depending on the model, a directed graph can be cyclic or acyclic.
Once a model has been created, the next task is to *describe an algorithm to identify optimal key regulators of a subgroup of the network.* Formally, the task can be defined using the following definitions.

Definition 3.1.2 The subgroup of a network whose key regulator is to be determined is known as a *set of hull level zero nodes.* This set is defined as a non-empty set of nodes N_0, where $N_0 \subset N$ and $N_0 \neq \emptyset$.

A node connected to any of the nodes in hull level zero is considered to be in hull level 1; consequently a node connected to a hull level 1 node is considered to be in hull level 2, and so on. Thereby to identify the hull level of a node, for each graph a hull determining function h can be defined as below.

Definition 3.1.3 The *hull function h* maps the nodes to their maximal hull level; thus $h(n) = c$, where $n \in N$, $c \in \mathbb{Z}^*$ and $0 \leq c \leq |N|$, since the maximum number of hull levels is equal to the number of nodes in N. If a node is in more than one hull level, then the function returns the maximum value.

Definition 3.1.4 A *key regulator* k is a node capable of regulating all hull level zero nodes, where $k \in N$. It is defined as a node with $N_0 \subseteq R_k$. A set of all possible key regulators is K, where $K \subseteq N$ and $\forall k \in K : k$ is a key regulator.

According to the specifications of the model, there are various criteria for choosing optimal key regulators from the set of all possible key regulators K.

Definition 3.1.5 An *operator* $\preceq$ is defined as a binary comparison operator specific for each network model. It is used to compare between the possible key regulators and identify optimal ones amongst them.

Definition 3.1.6 An *optimal key regulator* k' for this approach, where $k' \in K$, is defined as a key regulator node with $N_0 \subseteq R_{k'}$ and $\forall k \in K \rightarrow k \preceq k'$. Since more than one optimal key regulator may exist, thereby a set K' is a set of all possible optimal key regulators, where $K' \subseteq K \subseteq N$.

3.2 General Approach

Given a complete network and the target regulated group, it can be noticed that the problem exhibits *optimal substructure*. Optimal substructure means that optimal solutions of sub-problems can be used to find the optimal solutions of the overall problem. As such, taking a network of reduced hull levels and finding the key regulator for it, then the key regulator of the complete network is a regulator connected to the one of the reduced network. Therefore the problem at hand exhibits optimal substructure.

When the problem can be decomposed into sub-problems and a recursive algorithm revisits the sub-problems repeatedly; then these can be considered as *overlapping sub-problems.* For the problem at hand; at any point, to check if a node (at a high hull level) is a key regulator, then a check must be performed on whether the nodes it is connected to (at lower hull levels) regulate the hull zero nodes; thus the check is repeated for each of the lower hull level nodes. Therefore the sub-problems overlap.

Since the problem exhibits the properties of optimal substructure and has overlapping sub-problems, then it can be solved using a *dynamic programming approach*, thereby taking less time than naive methods. The dynamic programming approach is a *bottom-up approach* where all sub-problems that might be needed are solved in advance and then used to build up solutions to larger problems. The first hull level is considered first and the regulations of its nodes are stored. Then the nodes in increasing hull levels are propagated through, using the solutions to the previous sub-problems for the current investigated node. Once the whole network has been propagated through, then it is possible to identify all possible key regulators and then optimise by distinguishing the most optimal ones.

Before propagating through the hull levels, initializations of the R sets must be made and the connections of the nodes in hull level zero must be analyzed. This is necessary because nodes in hull level zero could be connected to one another. The overall procedure can be summarised in the steps shown in algorithm 1, where the exact implementation of the 3 functions is modified to fit the different network models investigated.

Algorithm 1 GetKeyRegulators()

```
1: InitNodeInfo()
2: for all $n \in N_0$ do                ▷ Initialization of connections of hull zero nodes
3:   for all $< n, i > \in E \wedge i \in N_0$ do
4:     UpdateNodeInfo($n, i$)
5:   end for
6:   for all $< i, n > \in E \wedge i \in N_0$ do
7:     UpdateNodeInfo($i, n$)
8:   end for
9: end for
10: for $i : 1 \dots |N|$ do             ▷ Analyzing connections of all other nodes
11:   for all $n \in N \wedge h(n) = i$ do
12:     for all $< n, m > \in E$ do
13:       UpdateNodeInfo($n, m$)
14:     end for
15:   end for
16: end for
17: $K' =$ GetOptimalKeyRegulators()
18: return $K'$
```

Where the general funcitonality of the 3 methods being called are:

1. *InitNodeInfo()*: used to perform the necessary initializations such as setting the initial R sets of all nodes

2. *UpdateNodeInfo(n,m)*: used to handle propagation of an interaction from node n to m, by combining the set of regulated nodes of m with those of n

3. *GetOptimalKeyRegulators()*: used to choose optimal key regulators by applying model specific criteria in choosing a key regulator

Chapter 4

Different Network Models

Presented in this section are the different models suggested for modelling gene regulatory networks and the key identifier algorithm implemented on each model. The first model is a most abstract one and then gradually these abstractions are removed to arrive at the closest possible model to real biological data at this logical level approach.

4.1 Acyclic unsigned network model

Model description

The first model for a gene regulatory network is an abstract one, modelling it as a *directed graph* as defined in definitions 3.1.1 to 3.1.6.
The abstraction is in handling the edges. An edge in this model is directed, yet has no label to indicate its function. From the biological point of view, this model uses nodes to represent genes and uses edges to represent the regulations between them; however the nature of the regulation is not indi-

cated. In this model, networks are assumed to be acyclic, thus biologically it contains no positive and negative feedback circuits.
For this model, the only criteria to distinguish between all possible key regulators is the hull level, where a key regulator having a higher hull level is defined to be more optimal. Thus the comparison operator in 3.1.5 can be redefined as follows.

Definition 4.1.1 The *operator* $\preceq$ for this model is defined as follows:

$$x, y \in N : x \preceq y = \begin{cases} \textit{true} & \textit{if } h(x) \leq h(y) \\ false & \textit{otherwise} \end{cases}$$

A graphical representation of such a model can be illustrated as in figure 4.1.

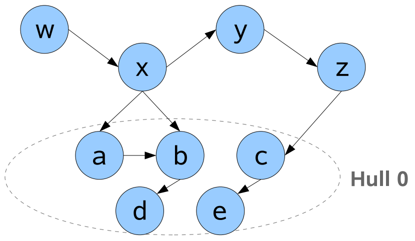

Figure 4.1: Gene Regulatory Network represented as a acyclic directed graph with non-signed edges.

Applying the stated definitions, the following information is obtained:

$N = \{a, b, c, d, e, w, x, y, z\}$,

$E = \{< a, b >, < b, d >, < c, e >, < w, x >, < x, a >,$

$< x, b >, < x, y >, < y, z >, < z, c >\}$,

$h(a) = h(b) = h(c) = h(d) = h(e) = 0$,

$h(z) = 1$, $h(y) = 2$, $h(x) = 3$, $h(w) = 4$,

$N_0 = \{a, b, c, d, e\}$.

Notice that although node x is in both hull levels 1 and 3 and node w in hull levels 2 and 4; the hull determining function h assigns only one hull level to the nodes.

Pseudo Code

In order to identify the optimal key regulators for this model, the same general algorithm described in algorithm 1 is followed, but the exact steps of the 3 functions being called by the algorithm is dependant on the network model. For this model, these functions are precisely defined as follows.

The following code fragment initializes the regulatory sets of the graph:

Algorithm 2 InitNodeInfo()

```
1: for all $n \in N$ do
2:     $R_n \leftarrow \{\}$;
3: end for
```

The following code fragment is used to handle regulation propagation through union operations:

Algorithm 3 UpdateNodeInfo(n,m)

```
1: $\hat{\mathrm{R}} \leftarrow \{\}$
2: if $m \in N_0$ then
3:     $\hat{\mathrm{R}} \leftarrow \{m\}$
4: end if
5: $R_n \leftarrow R_n \cup R_m \cup \hat{\mathrm{R}}$;
```

After the R sets of all nodes have been set, the task of identifying the key regulator node is quite simple. As defined earlier, any node n can be a key regulator node if it satisfies the following condition: $N_0 \subseteq R_n$, or since the R sets only contain N_0 nodes then more specifically: $N_0 = R_n$; thus $\forall x \in N_0 \rightarrow x \in R_n$.

Due to biological reasons, a key regulator is defined as more optimal if it is at a higher hull level. Thus the 3rd function specified earlier which identifies all optimal key regulators follows:

Algorithm 4 GetOptimalKeyRegulators()

```
1: for all $n \in N$ do                                ▷ Setting $K$
2:     if $R_n = N_0$ then
3:         $K \leftarrow K \cup \{n\}$
4:     end if
5: end for
6: $K' \leftarrow \{\}$
7: for all $n \in K$ do                                ▷ Setting $K'$
8:     allSmaller ← true
9:     for all $m \in K \wedge n \neq m$ do
10:        if $m \npreceq n$ then
11:            allSmaller ← false
12:        end if
13:    end for
14:    if allSmaller = true then
15:        $K' = K' \cup \{n\}$
16:    end if
17: end for
18: return $K'$
```

Applying the previous algorithm on the network in figure 4.1, then $K = \{w, x\}$ but $K' = \{w\}$ since $x \preceq w$ and $w \npreceq x$.

4.2 Acyclic signed network model

Model description

The previous model can be modified to include the semantics of the relations between the nodes. As previously stated, the interactions between genes can be either *activations* (+) or *repressions* (−). Thus a new model will be constructed for gene regulatory networks. This model still defines them as a *directed graph* as earlier in 3.1.1. However for each graph a new function g is introduced which determines the action of an edge.

Definition 4.2.1 The *edge function* g is a function that determines the nature of an edge. It is defined as $g : E \rightarrow A$, where A is a set indicating the possible actions of the edge, thus $A = \{+, -\}$. Thus for an edge $e \in E$ where $e =< i, j >$ and $i, j \in N$, then $g(e) = +$ means that node i has an activating action on node j and $g(e) = -$ means node i has an repressing action on node j.

Since a node can up regulate or down regulate another node, it is not sufficient to state a node has a set R of all the nodes it regulates; in contrast a distinction has to be made between the nodes that it activates and those that it represses. This need necessitates redefining the properties of N as follows.

Definition 4.2.2 The set N is defined as a set of *nodes,* where each node $n \in N$ has two sets: R^+ and R^-, where $R = R^- \cup R^+$, R_n^+ is the set of the nodes which are up regulated (or activated) by n and R_n^- is the set of the nodes which are down regulated (or repressed) by n.

If a node x up regulates another node y and $y \in N_0$, then it will have the same up and down regulating function of that node; $R_x^+ = R_x^+ \cup R_y^+ \cup \{y\}$ and $R_x^- = R_x^- \cup R_y^-$. However if a node x down regulates another node y, then this means that it has a reverse function to that of the regulated node. In simpler terms, this means that x can up regulate what y down regulates and vice versa, thus $R_x^+ = R_x^+ \cup R_y^-$ and $R_x^- = R_x^- \cup R_y^+ \cup \{y\}$. Note that if $y \notin N_0$ then the same rules apply but without including $\{y\}$ in the relations. Hull level 0 nodes are defined as in 3.1.2. Since $R = R^+ \cup R^-$, then the same definition of a key regulator as in 3.1.4 holds.

In order to choose the most optimal key regulators from all possible ones, the same $\preceq$ operator as defined in definition 4.1.1 is used. Thereby an optimal key regulator is defined as earlier in definition 3.1.6.

A graphical representation of such a model can be illustrated as in figure 4.2. Applying these new definitions applies the same sets as stated earlier.

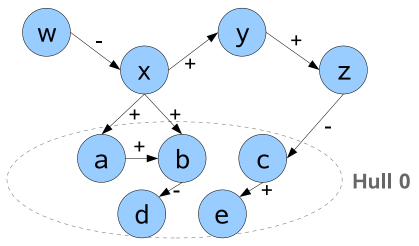

Figure 4.2: Gene regulatory network represented as an acyclic directed graph with signed edges.

Pseudo Code

In order to identify the key regulator in such a network the same general algorithm 1 is followed, however the functions in algorithms 2 and 3 have to be modified to satisfy the new model specifications. The $GetOptimalKey()$ function in algorithm 4 remains the same, since $R = R^+ \cup R^-$, so the check condition in line 3 is still applicable.

The following code fragment initializes the regulatory sets of the graph:

Algorithm 5 InitNodeInfo()

```
1: for all $n \in N$ do
2:     $R_n^+ \leftarrow \{\}$;
3:     $R_n^- \leftarrow \{\}$;
4: end for
```

The following code fragment is used to handle regulation propagation through union operations:

Algorithm 6 UpdateNodeInfo(n,m)

```
1: $\hat{R} \leftarrow \{\}$
2: if $m \in N_0$ then
3:     $\hat{R} \leftarrow \{m\}$
4: end if
5: if $g(<n, m>) = +$ then
6:     $R_n^+ \leftarrow R_n^+ \cup R_m^+ \cup \hat{R}$;
7:     $R_n^- \leftarrow R_n^- \cup R_m^-$;
8: else
9:     $R_n^+ \leftarrow R_n^+ \cup R_m^-$;
10:    $R_n^- \leftarrow R_n^- \cup R_m^+ \cup \hat{R}$;
11: end if
```

Applying the overall algorithm on the network in figure 4.2, then $K = \{w, x\}$, where $R_w^+ = \{d, e\}$, $R_w^- = \{a, b, c\}$, $R_x^+ = \{a, b, c\}$, $R_x^- = \{d, e\}$ but $K' = \{w\}$ because $x \preceq w$, $w \npreceq x$, and the $\preceq$ operator is defined only with respect to the hull levels.

4.3 Cyclic signed network model

Model description

The next step is to take the model one step further by raising the assumption that the network is acyclic, and enabling the graph to have *cycles*. As discussed earlier, cycles have several biological effects in the model; whether the cycle is a positive or a negative feedback loop and whether an edge is activating or repressing it, affects the method to finding the key regulator. The definitions for this cyclic model are the same as the previous one (3.1.1 to 4.2.2), except that the graph is now allowed to have cycles. A cycle is a positive or negative feedback cycle depending on the number of negative edges within it.

Definition 4.3.1 A *positive feedback cycle* is one where there is an even number of negative edges [14].

Definition 4.3.2 A *negative feedback cycle* is one where there are an odd number of negative edges [14].

An example of such a model is shown in figure 4.3 below. This network consists of 3 cycles; the cycle u, v, x and the self-cycle y are negative feedback loops because they each have 1 (odd) negative edge, and the cycle w, z is a positive feedback loop because it has 0 (even) negative edges.

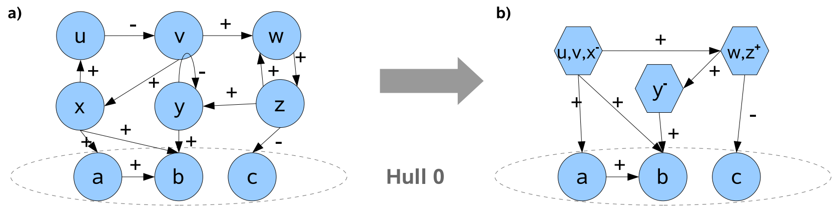

Figure 4.3: (a) Gene regulatory network represented as a signed directed graph with loops. (b) Same network after applying SCC algorithm, showing the cycles collapsed as new nodes with superscripts indicating the cycle type.

Handling cycles: Strongly Connected Components Algorithm

Introducing cycles into the model creates many problems when trying to apply the key identifier algorithm in 3.2. A main problem is assigning the hull levels correctly. Assigning of hull levels was previously done by starting at the regulated hull zero level and then moving outwards incrementing one level with each edge. With cycles, one starts to go in a loop indefinitely and keeps assigning higher hull levels with each traversal through a cycle. Consequently, there is no upper bound on the number of hull levels.

Being unable to assign hull levels hinders the general algorithm proposed in 3.2. The solution proposed to this problem is to collapse cycles into a single node. This idea is based on the *Strongly Connected Components (SCC) algorithm* in [5]. This algorithm collapses all cycles within a graph into single nodes, thus leaving an acyclic graph. The algorithm is based on the *Depth First Search (DFS)* [5]. Thereby it is more convenient to state the DFS algorithm before going on with the SCC one.

The *depth first search algorithm* is a traversal algorithm on a graph, aiming at traversing all the nodes within a graph but in a depth-first manner.

It starts by selecting some node and explores as far as possible along each undiscovered branch before backtracking. If any node remains undiscovered, the search is repeated from this node. The procedure repeats till all nodes have been discovered. While performing the traversal, the algorithm marks each node with a discovery time, a finishing time and a colour. The discovery time denotes the time that the node is discovered during the search; it is turned grey at this stage. The finishing time denotes the time in which all of the node's descendants have been discovered; it is turned black at this stage. All nodes are initially white, denoting that they haven't been discovered yet. Therefore at any instant in the algorithm, the colour of the node denotes which stage it is in.

The *strongly connected components algorithm* is composed of four main steps:

1. Perform depth first search to compute the finishing time of each node.
2. Compute transpose of the graph (i.e. change the direction of all edges).
3. Perform depth first search on graph from 2 but consider the nodes in order of decreasing finishing time (as calculated in step 1).
4. When performing step 3, whenever a white colour node is chosen, all the nodes traversed following it till it is reached again are considered to be components of a new node representing a strongly connected component.

Therefore before performing the general key identifier algorithm (3.2) on this model, first the strongly connected components algorithm is run on the network in order to collapse cycles and loops into single nodes.

However, the model contains signed edges, thus simply collapsing the cycles into new nodes would not suffice to solve the problem. Alternatively, a distinction must be made between the different types of cycles during the 4th step.

As discussed earlier, there are two types of loops depending on the count of negative signs within a cycle; namely *positive feedback cycles* and *negative feedback cycles*. Thus when collapsing a cycle into a node, a count should be performed on the number of negative signs within the cycle; if this count is even then the new node is a *positive SCC*, otherwise it is a *negative SCC*. For the network in figure 4.3, the cycles u, v, x and y each contain one negative edge thus their collapsed node is a negative SCC, on the other hand the cycle containing w, z contains zero negative arcs so it is a positive SCC.

Definition 4.3.3 A *function c* is defined as a function that determines whether a node is a normal node or representing positive or negative collapsed components; $c : N \to C$, where $C = \{normal, positive, negative\}$.

After collapsing the nodes into new ones, some processing will be needed to produce the new *edge tuples* E, which connect the new nodes with the old ones and vice versa. Multiple edges between collapsed nodes and old nodes may exist.

Pseudo code

The general algorithm in 3.2 is still followed, however a pre-processing function that collapses cycles is performed at the start. The resultant graph is an acyclic one but with two new types of nodes introduced; *positive SCC* and *negative SCC* nodes. The challenge faced now is handling edges regulating these new nodes, which will be discussed next after defining the regulatory strength.

Further quality Criteria

Criteria I: Regulatory Strength

If a regulator is connected to a node by many different connections, then this means that it has a stronger effect on that node. In certain applications, a node having numerous distinct paths to the regulated subgroup maybe more desirable than one with fewer paths. The reason for this is that the regulation will be more robust against disturbances. For example, consider the regulation from node a to node b having two paths, one through node c and another through node d. If node c is repressed or deactivated then there is still another regulatory path through node d, to which a can effect b. In other words, node s has twice the regualtory power on node b.
To handle this, the idea is to modify the model such that for each node in the lists R^+ and R^-, a *regulatory strength* (defined below in 4.3.4) is computed. When a node is to be regulated twice then its regualtory strength is increased by a factor of 1.

Definition 4.3.4 The *regulatory strength,* $s^n_{R_m}$, of a node n in a node set R_m where $n, m \in N$ is defined a representation of the different regulation paths from m to n and the regulatory power of these paths. The effect of the type of nodes on the regulation paths is given by the multiplier function defined below.

As discussed in the background chapter, *negative feedback loops* are more controlled components. The effect of activating such a loop leads to a decayed activation of nodes. Since within the models discussed so far, the time factor was not taken into consideration, therefore such a decayed activation can be handled by giving it a lower regulation power than that of a normal activation.

Consequently, if the node represents a negative feedback loop then the regualtory strength is affected by a factor of 0.5. This abstraction simply indicates that a negative feedback loop has been encountered. On the other hand, repressing a negative feedback loop has no biological meaning. Thus such an action is treated as a normally repressed node; thereby inverting the R^+ and R^- lists of the negative SCC nodes and merging them with its regulator.

From a similar biological point of view, *positive feedback loops* have a dynamo effect. Activating them gives them the chance to start activating continually. To model this without time, the activation of such a node a given a strength of 2.0; thereby denoting that activating such a node gives greater regulation power and is similar to having two different activating pathways leading to this node. Repressing a positive feedback loop has no significant biological meaning, since it does not enhance any special behaviour. Thus again such a repression to a positive SCC node is treated as a normal repression to a node.

The effect of these abstractions and the interpretation of multiple regulations on the regualtory strength is given by the multiplier function defined below.

Definition 4.3.5 A *multiplier function* p is defined as a function returning the power of a regulation according to the type of the node and whether the edge is activating or repressing it; where formally $p : N \times N \rightarrow \mathbb{R}^+$. For this model, $\forall <i, j> \in E$:

$$p(i,j) = \begin{cases} 0.5 & \textit{if } c(j) = \textit{positive} \quad \wedge \quad g(<i,j>) = + \\ 2.0 & \textit{if } c(j) = \textit{negative} \quad \wedge \quad g(<i,j>) = + \\ 1.0 & \textit{otherwise} \end{cases}$$

In order to perform the correct propagation of the regulatory stength factors, they should be initialized using the following rules (in order):

$$\forall x, y \rightarrow s^x_y = 0$$

$$\text{then}$$

$$\forall n \in N_0 \rightarrow (s^n_{R^+_n} = 1 \wedge s^n_{R^-_n} = 1)$$

These rules should be inserted in the initialization algorithm (5), after the 4th line.

Handling the collapsed components has an effect on the update function (6) in 4.2. The following code lines are added to it after line 11:

Algorithm 7 additional code to update algorithm 6

```
1: for all $i \in R^+_n$ do
2:     $s^i_{R^+_n} = s^i_{R^+_n} + (s^i_{R^+_m} * p(n, m))$
3: end for
4: for all $i \in R^-_n$ do
5:     $s^i_{R^-_n} = s^i_{R^-_n} + (s^i_{R^-_m} * p(n, m))$
6: end for
```

As an example, the strength will be calculated for the network in figure 4.3 (b).

According to the general algorithm in 3.2, the initializations are made then the connections of hull zero nodes are taken into consideration. So after the full initialization, the following are obtained:

$n \in N_0$	R_n	$s^a_{R_n}$	$s^b_{R_n}$	$s^c_{R_n}$
a	$R_a^+ = \{b\}$	1	1	0
	$R_a^- = \{\}$	1	0	0
b	$R_b^+ = \{\}$	0	1	0
	$R_b^- = \{\}$	0	1	0
c	$R_c^+ = \{\}$	0	0	1
	$R_c^- = \{\}$	0	0	1

Then after continuing the algorithm till the end, a similar table can be produced showing the strength of all other nodes and the method of their calculation:

$n \notin N_0$	R_n	$s^a_{R_n}$	$s^b_{R_n}$	$s^c_{R_n}$
y	$R_y^+ = \{b\}$	0	$0 + 1 * 1 = 1$	0
	$R_y^- = \{\}$	0	0	0
{w,z}	$R_{w,z}^+ = \{b\}$	0	$0 + 1 * 0.5 = 0.5$	0
	$R_{w,z}^- = \{c\}$	0	0	$0 + 1 * 1 = 1$
{u,v,x}	$R_{u,v,x}^+ = \{a, b\}$	$0 + 1 * 1 = 1$	$0 + 1 * 1 + 1 * 1 + 0.5 * 2 = 3$	0
	$R_{u,v,x}^- = \{c\}$	0	0	$0 + 1 * 2 = 2$

Using the newly calculated strength, it is possible to modify the comparator operator $\preceq$ in 4.1.1 to involve comparing the regulatory strength of the regulated node, where a key regulator is favoured if it has a higher strength of the regulated nodes. However, the exact implementation of this optimization is application specific and thus will be left as an open topic.

Criteria II: Uncertainty Factor

After calculating the regulations of a node, it is possible to find a node capable of both up regulating and down regulating another node. This happens because there are two regulatory paths from the node to other one in which one leads to an activation and the other to a repression. Thereby the need arises to introduce an *uncertainty factor* for each node as follows:

Definition 4.3.6 An *uncertainty factor* U_n of a regulated node n is a value denoting the unsureness of the regulatory power of a node to the hull zero nodes (N_0); where $U_n = |R_n^+ \cap R_n^-|$ and the $\cap$ operation produces a unique set of distinct nodes.

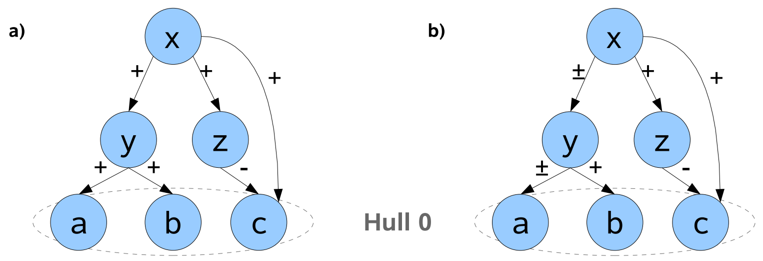

Figure 4.4: (a) Gene regulatory network represented as an acyclic directed graph with signed edges (b) The same gene regulatory network but including dual $\pm$ edges.

As an example, consider the graph in figure 4.4 (a). The following can be inferred:

$$R_y^+ = \{a, b\},\ R_y^- = \{\},\ R_z^+ = \{\},\ R_z^- = \{c\}$$
$$R_x^+ = \{a, b, c\},\ R_x^- = \{c\}$$

Thus the uncertainty factor for node x: $U_x = |R_x^+ \cap R_x^-| = |\{c\}| = 1$.

In addition, after inspecting real biological networks, an observation was made about the edges in the graph. Edges showed repressions ($+$) and activations ($-$) but there were also edges with *dual functionality* ($\pm$). These edges affect the uncertainty factor calculated above and also affect the network model and the handling of edges throughout the key identifier algorithm in 3.2.

The model description has to modified since the alphabet of the possible edge actions defined for edge determining function g in definition 4.2.1 has to be changed to: $A = \{+, -, \pm\}$.

Biologically, the behaviour of these types of dual edges is uncertain; over a period of time their functionality may change. The application might favour considering such uncertain edges as only one action if it is maximizing that action. Therefore the handling of such edges is application specific.

To handle dual edges, a check is performed at the beginning of the algorithm to know the user's preference of handling (i.e. whether to handle them as activating, repressing or both). Then in the code fragment concerning updating the node information (algorithm 6), an extra if branch is inserted. If the preference is to consider them as dual functionality then both $+$ and $-$ branches are performed, otherwise the branch according to the preference is followed.

As an example consider the graph in figure 4.4 (b), which the same as that in (a) but with dual $\pm$ edges added. Considering the dual edges at $+$ ones yields the results discussed earlier on figure 4.4. Whereas considering them as $-$ edges, yields the following:

$$R_y^+ = \{b\},\, R_y^- = \{a\},\, R_z^+ = \{\},\, R_z^- = \{c\}$$
$$R_x^+ = \{a, c\},\, R_x^- = \{b, c\}$$

Considering them as $\pm$ edges, i.e. as both $+$ and $-$, yields the following:

$$R_y^+ = \{a, b\},\, R_y^- = \{a\},\, R_z^+ = \{\},\, R_z^- = \{c\}$$
$$R_x^+ = \{a, a, b, c\},\, R_x^- = \{a, a, b, c\}$$

As concerning the calculation of the uncertainty factor, the algorithm is traversed while specifically choosing to handle dual edges as both, then the calculation remains as stated in definition 4.3.6. So if the application required a different handling than that favouring duality, first the algorithm is traversed to calculate the uncertainty factors of all nodes then the algorithm is traversed again according to the application preference. Applying this to the previous example of figure 4.4 (b):

$$U_y = 1,\, U_z = 0,\, U_x = 3$$

Chapter 5

Implementation

In order to implement the models and the key identifier algorithms, and to test them on sample and real biological data, an implementation was made using JAVA as the programming language. In particular Java SE with JDK 5.0 was used.

For each network model, the input network is transformed into a graph data structure according to its respective definition. Then the main class performs any necessary pre-processing then it runs the respective key identifier algorithm. The output produced is a model of the network, with all possible optimal key regulators.

The input to the model is given as an external file. For testing purposes randomly generated networks were used given in the DIMACS format. This format was developed by the center for DIscrete MAthematics and theoretical Computer Science, which is a commonly used format to represent graphs. It consists of four types of statements:

1. *Comment lines: c* <the comment>

2. *Problem line:* $p < n >< e >$; stating the number of nodes $< n >$ and edges $< e >$

3. *Node lines:* $n < id >< name >$; listing each node with its id and name

4. *Edge lines:* $e < n1 >< n2 >< wt >$; where $< n1 >$ is the id of the starting node and $< n2 >$ is the id of the end node, $< wt >$ is the weight of the edge. Since edges are activating, repressing or both then the weight will be +,- or ? respectively.

A converter method was developed to convert real network data given in other formats to the DIMACS format. In addition, the set of hull zero nodes were given as an external input in a file, which simply includes a listing of the identifiers of the nodes. Another converter method was developed to transform a real representation of hull zero nodes into a standard one accepted by the general program.

The program is made to output a representation of the model including hull zero nodes and optimal key regulator nodes. The output file is written in the DOT language; which is a plain text graph description language. It is similar to the DIMACS format, but more attributes regarding the graphical appearance of the graph can be specified. The advantage of having the output as a DOT file, is that there exists a *dot* program, part of the Graphviz package [15], which is capable of reading attributed graph text files and producing a graphical visualization of the directed graph as a hierarchy.

Chapter 6

Results

6.1 Real Data

Many examples of gene regulatory networks are available in Biological databases. The *KEGG (Kyoto Encyclopedia of Genes and Genomes)* [10] is a knowledge base of biological systems, consisting of the genetic building blocks of genes and proteins, chemical building blocks of substances, molecular wiring diagrams of interaction and reaction networks, and the hierarchies and relationships of various biological objects. A part of KEGG known as *KEGG PATHWAY*, offers the molecular interactions and reaction networks for metabolism, various cellular processes, and human diseases. Amongst them are many signal transduction networks which can be modelled using the proposed model and then identify the key regulator for given subgroups in them.

After examining the networks available on the KEGG database, they were found to be simple, signed networks. Ignoring the signs on the edges, they can be modelled them using the first model presented. Then taking them signed as given, the second presented model can be used.

Another biological database is *RegulonDB* [13]. It is a database containing models of the complex regulatory networks of the cell. To the interest of this work, this database offers a dataset of Escherichia coli (E. coli); which is one of many species of bacteria living in the lower intestines of mammals. The database offers the regulatory network interactions of transcription factors and genes. The network can be modelled as a three state cyclic signed network; thus providing the test data for the third model. However some edges have unknown functionality (?), and these interactions should be ignored. Moreover incomplete data and unreal data such as phantom genes should be removed from the input. It should be noted that this database only consists of 20% of the regulatory transcriptions in the cell [11].

Many gene families have been identified in E. coli; each of which has a certain function. Thus given these families, they can be used as a group of hull zero nodes to which a key regulator is required. A list of the gene families is found in the *GenProtEC (E. Coli genome and proteome)* [6] database. This database is dedicated to the functions encoded by the E. coli.

6.2 Results Produced

For each network model, its respective real data was input to the program and a dot file showing the optimal key regulator nodes, the hull zero nodes and the nodes between them was output. Then using *dot*, a graphical representation of the output files was generated.

Input

For the third network model, an input network was obtained from RegulonDB [13] , showing the regulatory interactions of E. coli. A graphical representation of the complexity which is given to the program is shown below. This figure is given here only to provide an impression of the complexity of the input regulatory network. However, for implementation purposes a textual representation of the graph is used, in order to obtain the details of all the gene interactions.

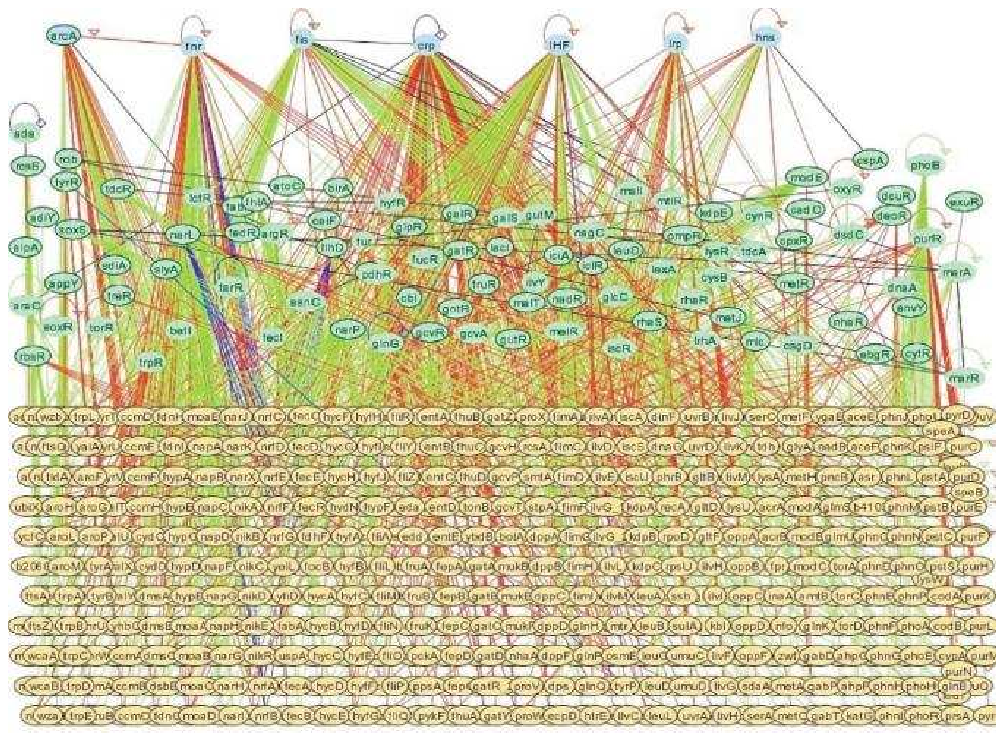

Figure 6.1: Overview of the transcriptional regulatory network in E. coli. Regulated genes are shown as yellow ovals, transcription factors are shown as green ovals and transcription factors considered to be global regulators are shown as blue ovals. The green lines indicate activation, red lines indicate repression and dark blue lines indicate dual regulation (activation and repression) [11].

After running the pre-processing function on the input data, it was found to contain 1346 nodes, 53 positive cycles and 31 negative cycles. Thereby collapsing these cycles into new nodes was a necessary step.
Since the data obtained from the RegulonDB is not entirely complete, there are some family member genes (as obtained from the GenProtEC database) which are not present in the network data. However a few families were chosen which have more than 70% of their members available in the network data. The GNTP family is one of these families.
From the GenProtEC database [6], it was possible to extract the following information about the GNTP family, and hence the set of N_0 nodes of known:

Gene	Module	Function
dsdX	b2365	transport protein
ygbN	b2740	putative transport protein
gntT	b3415	high-affinity gluconate permease in GNT I system
idnT	b4265	L-idonate transport protein
yjhF	b4296	KpLE2 phage-like element; putative transport protein
gntP	b4321	gluconate transport protein, GNT III system
gntU	b4476	split gene, low-affinity gluconate transport permease protein in GNT I system

Output

The program is run on the input data and produces a DOT file, which contains a text description of network graph containing a distinction of the hull zero nodes and the optimal key regulators. Since the original E. coli network is very large, the program is made to produce a reduced graph only including the nodes involved in the regulations of the chosen subgroup. The dot file is then converted by *dot* program to produce a graphical image of the network as seen below.

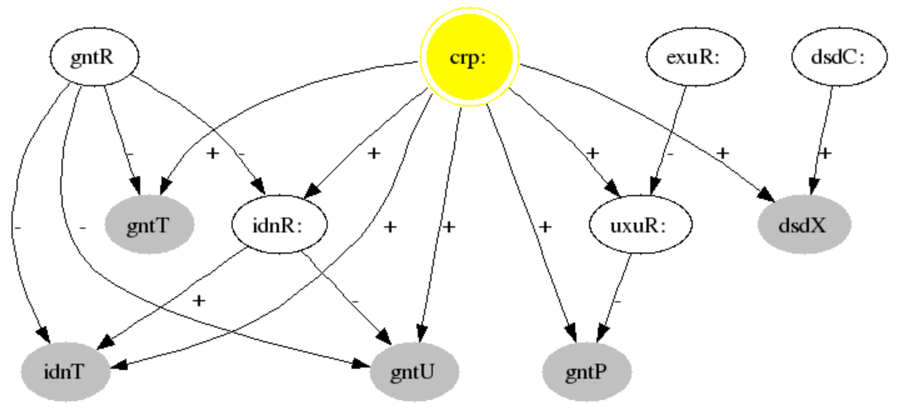

Figure 6.2: A reduced E. coli network produced by *dot* showing optimal key regulators of the GNTP family and only those nodes included in the regulations. The key regulator is shown in a big yellow circle, hull zero nodes are in grey, negative SCC components are written in blue and positive SCC components are written in red.

In addition the program produces a summary file containing a list of the optimal key regulators of the GNTP family, with the regulatory strength of each node regulated by the key regulators. This output file is as follows; where the number in the square brackets for each node is its regulatory strength.

```
Key regulator is node:  crp:
with R+:  idnT[x1.5] gntU[x1.0] gntT[x1.0] gntP[x1.0] dsdX[x1.0]
with R-:  gntP[x2.0] gntU[x0.5]
```

From the output graph in figure 6.2, it is easy to verify that the calculation of the regulatory strength as discussed previously.
Biologically, it is known that the *crp* gene is a global or key regulator in E. coli as mentioned in [11, 17]. This verifies the output produced.

Chapter 7

Conclusion and Open Topics

With the advancements in technology, the amount of biological data obtained is plentiful. Thus the field of bioinformatics has emerged in which computer systems are used to analyze the data and extract significant information. In particular, due to the increase in genome sequencing projects, there has been an increase in the amount of data relating to the interactions leading to the expression of genes. These interactions form the basis of essential processes like signal transduction, cell metabolism and embryonic development. They are represented as gene regulatory networks, which can be modelled as directed graphs where the nodes represent the genes and the edges represent the interactions between them. The networks might contain positive feedback cycles which when activated produce a dynamo regulating effect. Moreover, they might contain negative feedback cycles which when activated help in bringing activation levels back to normal. Therefore, modelling the networks will help to provide an understanding of the complex relations. In addition it would be possible to identify key regulators of cell responses which could induce an entire complex developmental pathway within an organism.

In this thesis, several models have been made for gene regulatory networks handling them as directed graphs and using the binary approach in which genes can only be activated or repressed. Three models were presented; the first one was acyclic and contained interactions without mentioning the type of the interaction. Then the second model considered acyclic graphs with activating and repressing edges. Lastly the third model, modelled them as cyclic graphs with interactions. For each model, a key identifier algorithm was implemented, which distinguishes nodes capable of regulating given subgroups of the network.

Further optimization criteria were introduced, thereby generating important properties like the regulatory strength of a regulated node and the uncertainty factor. These factors were not integrated into the process of choosing an optimal key regulator; however this is left as an open topic since it is dependant on the specifications of the application. Also remaining as an open topic, is the modelling of *dimer* molecules. Some genes are present as part of a two-component system known as dimers. Activating only one of them will not result in an activation of the system; both entities must be activated [12]. In this work, these were handled by treating each of the components separately. However, a better handling is needed to model the actual functionality of the dimer molecules. Another open topic is definition of the multiplier function 4.3.5 in the handling of cycles. The function used was simply an abstraction; dependant on the application it may be necessary to consider additional information in calculating the regulation power, such as the complexity of the collapsed node.

Bibliography

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter. *Molecular Biology of the Cell.* Garland Science, fourth edition, 2002.

[2] E. R. Alvarez-Buylla, M. Benítez, E. B. Dávila, Á. Chaos, C. Espinosa-Soto, and P. Padilla-Longoria. Gene regulatory network models for plant development. *Current Opinion in Plant Biology*, 2006.

[3] A. R. Borneman, J. A. Leigh-Bell, H. Yu, P. Bertone, M. Gerstein, and M. Snyder. Target hub proteins serve as master regulators of development in yeast. *Genes & Development*, 20(4):435–448, February 2006.

[4] P. Clote and R. Backofen. *Computational Molecular Biology: An Introduction*, pages 1–20. John Wiley and Sons Ltd, 2000.

[5] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*, pages 463–488. MIT Press, 1990.

[6] GenProtEC database: Genes and Proteins of Escherichia coli K-12 (GenProtEC). *http://genprotec.mbl.edu/prot_grp.php.*

[7] H. de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.

[8] T. S. Gardner and J. J. Faith. Reverse-engineering transcription control networks. *Physics of Life Reviews*, 2:65–88, March 2005.

[9] J. Hasty, D. McMillen, F. Isaacs, and J. J. Collins. Computational studies of gene regulatory networks: in numero molecular biology. *Nature Reviews Genetics*, 2(4):268–279, April 2001.

[10] M. Kanehisa, S. Goto, M. Hattori, K.F. Aoki-Kinoshita, M. Itoh, S. Kawashima, T. Katayama, M. Araki, and M. Hirakawa. From genomics to chemical genomics: new developments in kegg. *Nucleic Acids Research*, 34:D354–357, 2006. *http://www.genome.jp/kegg/*.

[11] A. Martínez-Antonio and J. Collado-Vides. Identifying global regulators in transcriptional regulatory networks in bacteria. *Current Opinion in Microbiology*, 6(5):482–9, 2003.

[12] D. J. Rodda, J. Chew, L. Lim, Y. Loh, B. Wang, H. Ng, and P. Robson. Transcriptional regulation of nanog by OCT4 and SOX2. *The Journal of Biological Chemistry*, 280(26):24731–7, 2005.

[13] H. Salgado, S. Gama-Castro, M. Peralta-Gil, E. Díaz-Peredo, F. Sánchez-Solano, A. Santos-Zavaleta, I. Martínez-Flores, V. Jiménez-Jacinto, C. Bonavides-Martínez, J. Segura-Salazar, A. Martvnez-Antonio, and J. Collado-Vides. RegulonDB (version 5.0): Escherichia coli k-12 transcriptional regulatory network, operon organization, and growth conditions. *Nucleic Acids Res*, 34(Database issue), January 2006. *http://regulondb.ccg.unam.mx/data/NetWorkSet.txt*.

[14] L. Segal. Chapter DN: Discrete Networks of Genes and Cells. Notes, October 2002.

[15] Graph Visualization Software. *http://www.graphviz.org/*.

[16] *http://www.ncbi.nlm.nih.gov/books/bookres.fcgi/mboc4/ch6f2.gif*.

[17] D. Thieffry, A. Huerta, Pérez-Rueda E., and J. Collado-Vides. From specific gene regulation to genomic networks: A global analysis of transcriptional regulation in escherichia coli. *BioEssays*, 20(5):433–440, 1998.