USE OF APRIORI KNOWLEDGE ON DYNAMIC BAYESIAN MODELS IN TIME-COURSE EXPRESSION DATA PREDICTION

Gokhul Krishna Kilaru

Submitted to the faculty of the School of Informatics
in partial fulfillment of the requirements
for the degree of
Master of Science in Bioinformatics
Indiana University

December 2011

Dedicated to the Almighty

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

No scientific endeavor is a result of an individual effort. Gratitude is beyond words… as I devotionally dedicate this piece of work to the Lord Almighty whose blessings are the pillars of my success. I sincerely thank all those who have directly or indirectly made this thesis possible.

First of all, I would like to express my deep and sincere gratitude to my advisor, Professor, Dr. Mathew J Palakal. Through his enthusiasm, inspiration and great efforts to explain things clearly and simply, Bioinformatics has always been fun for me. During the course of this thesis, the encouragement, sound advice and good teaching from him have helped me realize the full potential of thyself and the field.

With the heart-felt gratitude, I extol the motivation, wholehearted cooperation, altruistic guidance and constant help at every step of my thesis extended to me by my committee members, Dr. Malika Mahoui and Dr. Li Lin.

My sincere thanks to Dr. Meeta Pradhan, for her invaluable guidance at all times in the success of this work.

It gives me great pleasure in acknowledging the past and current members of TiMap lab, Deepali, Rini, Yogesh, Tulip and Kshithija for their constructive critiques.

Words are not enough to express wholehearted and affectionate gratitude to my beloved parents, Sri. Muddu Krishna Kilaru and Smt. Padmaja Krishna Kilaru, who stood like pillars of strength and support, and whose blessings have made me of what I am today.

As a whole I owe a lot to the School of Informatics at Indiana University, Indiana University Purdue University Indianapolis, for giving me this opportunity to pursue a bright career in the field of Bioinformatics.

Above all, this acknowledgement would be out of tune if I fail to mention people close to my heart, who stood by my side in sorrows and smiles alike and there is no match to the affection and encouragement given by them constantly. In this accord, I would like to mention my dearest friends and philosophers Aswani, Prathik Gadde, Prem, Prudhvi, Satish, Rahul Karmaai, Harish, Pavan, Sijin, Bullireddy, Sushant, Prathik Karnati, Namrita, Abhinita, Shreyas, Rohit, Anusha, Ram, Vinay, Gogi, Deepthi, Bhargav, Prasad and Mohan, who have been an immense source of energy, love and sense of humor during my graduation.

Last but not least, my heartfelt thanks to Mark McCreary, Mary, Jeff and all the administrative staff at the Indiana University School of Informatics, for their helping hand during the course of my graduation.

Any omission in this acknowledgement does not mean lack of gratitude.

ABSTRACT

Gokhul Krishna Kilaru

USE OF APRIORI KNOWLEDGE ON DYNAMIC BAYESIAN MODELS IN TIME-COURSE EXPRESSION DATA PREDICTION

Bayesian networks, one of the most widely used techniques to understand or predict the future by making use of current or previous data, have gained credence over the last decade for their ability to simulate large gene expression datasets to track and predict the reasons for changes in biological systems. In this work, we present a dynamic Bayesian model with gene annotation scores such as the gene characterization index (GCI) and the GenCards inferred functionality score (GIFtS) to understand and assess the prediction performance of the model by incorporating prior knowledge. Time-course breast cancer data including expression data about the genes in the breast cell-lines when treated with doxorubicin is considered for this study. Bayes server software was used for the simulations in a dynamic Bayesian environment with 8 and 19 genes on 12 different data combinations for each category of gene set to predict and understand the future time-course expression profiles when annotation scores are incorporated into the model. The 8-gene set predicted the next time course with $r>0.95$, and the 19-gene set yielded a value of $r>0.8$ in 92% cases of the simulation experiments. These results showed that incorporating prior knowledge into the dynamic Bayesian model for simulating the time-course expression data can improve the prediction performance when sufficient apriori parameters are provided.

1. INTRODUCTION

A great deal of research in the field of medicine has been resulting in identifying numerous diseases, disorders and their remedial treatments. The activity of the genes and their products in the biological pathways results in the biological phenomena (Efroni & Schaefer, 2007). Any minor changes or alterations in the equilibrium state of networks formed by these pathways results in complex disorders or more specifically diseases.

Understanding the complex disease networks is not an intuitive task. Many methodologies such as pathway construction, visualization and analysis have been helping researchers to understand and crack the minute changes that lead to a disease in any biological system.

Networks through which researchers understand the complex process vary with the genes of interest, expected results, hypothesis and more significantly the complexity of the process.

How does a researcher come to a conclusion or decides on which methodology to use? Well, the answer arises from the data that is being taken into consideration. High throughput experiments like microarray and mass spectrometry delivers large amounts of datasets that are hard to analyze manually. Development in the field of microarray technology leaves a vast resource of knowledge about the gene expression data, which is commonly used, in the field of bioinformatics to understand the complex biological systems' behavior (Kim & Miyano, 2004).

Bayesian networks, one of the most widely opted techniques to understand or predict the future by making use of the current or previous data, are gaining much importance since the last decade to simulate the huge gene expression datasets so as to

crack and predict the reasons for changes in the biological systems, which might yield remedial methodologies. The primary reason for relying on Bayesian networks is their capability to handle uncertainties in any given dataset. Understanding biological systems is completely uncertain, which calls for the support of Bayesian networks in real time through simulating biological data.

There are a wide variety of sub-techniques under Bayesian networks that can be used for different subjects like weather forecast, biology, politics, geography etc. One such part of Bayesian concept, which can be applied to the field of biology, is the Dynamic Bayesian Network (DBN).

DBNs make use of the time-course gene expression data when biological systems are being simulated. Time-course gene expression data serves as the backbone for any DBN simulation.

Gene expression varies at different time points in a biological system. Time-course gene expression data provides valuable information about those expression levels of a gene at that particular point of time. DBNs make use of the continuous feed of this time-course data and results in the knowledge of understanding the uncertainties in complex systems.

The activity of genes and their products in biological pathways results in biological phenomena (Efroni & Schaefer, 2007). Any minor changes in the equilibrium state of the networks formed by these pathways may result in complex disorders or diseases.

Understanding complex disease networks is not an intuitive task. Many methodologies such as pathway construction, visualization and analysis have been helping researchers to understand and crack the minute changes that lead to abnormalities in biological systems. Networks through which researchers understand the complex process vary with genes of interest, expected results, hypotheses, and more significantly, the complexity of the process. Development in the field of microarray technology provides a vast resource of knowledge about gene expression data, which is commonly used in the field of bioinformatics to understand complex biological systems' behavior (Kim & Miyano, 2004).

Annotation is a process through which the raw DNA sequences in a genome are assigned relevant information so as to classify the regions on the genome with respect to their functions (Silander, et al., 2006). Recent advancements in the field of database development and computational methods have allowed researchers to identify sources of annotation and develop a score to account for a gene's function and characteristics. Some of these sources include, but are not limited to, Genome Annotation Scores (GAS) algorithm, GO Annotation Quality (GAQ), the Gene Characterization Index (GCI), and GeneCards Inferred Functionality Score (Harel, et al., 2009).

Bayesian networks, one of the most widely used techniques to understand or predict the future state of a system by making use of current or previous data are gaining popularity. These network models can be used to simulate large gene expression datasets to track and predict the reasons for changes in biological systems, which might yield remedial methodologies. The primary reason for relying on Bayesian networks is their ability to handle uncertainties in a given dataset (Friedman, et al., 2000).

A wide variety of sub-techniques under Bayesian networks can be used for different areas such as meteorology, biology, politics, and geography (Abhik, Toyoaki, Peter, 2010), as demonstrated on *Jeopardy!* with the IBM computer Watson that makes use of Bayesian concepts. One such application area of Bayesian concept is in the field of biology where Dynamic Bayesian Network (Kim & Miyano, 2004) is used for simulating the current data to predict the future expression profiles of the genes. DBNs use time-course gene expression data when biological systems are being simulated. Time-course gene expression data serves as the backbone for any DBN simulation.

Genes' expression is different at different time points in a biological system. Time-course gene expression data provides valuable information about those expression levels of a gene at that particular point in time. DBNs use a continuous feed of this time-course data and results in an understanding of uncertainties in complex systems. One advantage of DBNs over traditional Bayesian networks is the ability to perform simulations in real time (Huang, et al., 2007). Gene knockout experiments permit researchers to understand biological system scenarios, when a particular gene is switched off. DBNs serve this purpose of analyzing a gene's activity when its neighbor gene is switched off along with the importance of the gene that is being switched off.

The end result of any gene expression in a given pathway is the activity level of another gene that comes down that particular pathway. As a whole, time-series simulation experiments using DBNs help in understanding the downstream target gene activity when a specific gene has been switched off or on. However, the challenges include understanding the organization and formation of these complex biological networks,

choosing the right gene in the model, declaring the node parameters in the DBNs, and deriving useful information from the end results.

This thesis presents a methodology for understanding how certain node parameters that account for a gene's annotation affect the DBN simulation model in predicting the next time-course gene expression data. This work also considers the design and architecture of a DBN model built on time-course gene expression data with additional properties of genes that are involved in a given disease.

1.1 Overview

Probabilistic models can be classified into four different categories (Singhal & Brown, 1997). They are Bayesian reasoning, evidence theory, robust statistics and recursive operators. When a system comprised of probabilities and uncertainties is considered, researchers usually shift their gears to Bayesian networks (BN), fuzzy logic and Hidden markov models. However, BNs represent the appropriate relative influences of real time facts. This is the primary reason for choosing BNs over issues involved with uncertainties such as gene expression data and their simulations.

Knowledge about different genes can be obtained simultaneously with the advancement of technology in recent years. And a heavy rise in the amounts of data is expected which stresses the need for understanding underlying patterns of the data, subjects of primary importance and eventual downstream targets.

Bayesian networks are special graphical models in which nodes represent the variables and their connections indicating the flow of information and node probability

distributions (Murphy, 2003 & Murphy, 1999). Bayesian networks are depicted in probabilistic directed acyclic graphs (DAG) (Spirtes, et al., 2001). The edge in a DAG points from one node called the parent node, to another node called the child node. If X1, X2 and X3 are the nodes in a network in such a way that X1 is the parent of X2 and X2 is the parent of X3, then X1 is the ancestor of X3, and X3 is called the descendant of X1. Each node in the network is conditionally independent from its non-descendants.

DBNs are the temporal extension of Bayesian networks, which are graphical models for relationships with probabilities among sets of variables. A DBN can be used to describe *causal* relations between variables in a probabilistic context. This means that a DBN can provide useful insight in the modeled reality by giving meaning to the interactions between variables for meaningful results. The random variables in a DBN do not have to be real numbers; they could just as easily be nominal values (e.g. male/female or true/false), ordinal values (e.g. grade or rank), or intervals (e.g. temperature $\leq 50^{\circ}C$, temperature $> 50^{\circ}C$).

A DBN model can be developed from expert knowledge or from a database using a combination of machine-learning techniques, or both (Moloshak, et al., 2002). These properties make the DBN formalism essential for the medical domain, as this domain has an abundance of both expert knowledge and databases of patient records. On the other hand, many interactions between variables in physiological processes are still unclear. More insight into these interactions can be gained by modeling them with a DBN. Creating models for temporal reasoning with DBNs sounds promising at first sight (Hulst, 2006), but many questions remain. For instance: *How do we obtain the structure and parameters? What is the performance of inference? How much data do we need for*

learning the parameters? If possible, how much does a DBN outperform a BN when applied to the same problem? (Hulst, 2006).

One major drawback of Bayesian network concept is the lack of regular cyclic mechanism, which is essential for gene regulation or expression activity in a biological system. DBNs rely completely on time-course data, which leads to data discretization into several classes (Heckerman, 2008). This discretization leads to information loss in the network, giving rise to uncertainties in the network. Simulating biological information using DBNs essentially involves training the Bayesian model to reach the desired uncertainties that a biological system usually exhibits. After identifying the desired uncertainties, the model is carefully observed for the changes on the other variables, which are essential and useful in identifying novel treatment methods.

In this study, we have incorporated the annotation scores such as the gene characterization index (GCI) and genecards inferred functionality score (GIFtS) which account for the gene's annotation status in the literature and also its functionality and characteristic. This study helps us to understand the effect of incorporating additional knowledge parameters as node attributes into the DBN model to understand the changes involved in the next time-point.

1.2 Goal of Research

The goal of this research is to understand and assess the prediction performance of the DBN model by incorporating prior knowledge. The results showed that incorporating prior knowledge into the dynamic Bayesian model for simulating the time-

course expression data could improve the prediction performance when sufficient apriori parameters are provided. The following steps are employed:

1. Extract time-course breast cancer gene expression data from Gene Expression Omnibus database.
2. In the data pre-processing steps, use Cyclic Loess normalization, MA plots, Box plots and Volcano plots to normalize, analyze and improve the data to fit in the DBN model.
3. Calculate the Pearson correlation among experiments and genes in the dataset.
4. Identify those experiments and the genes that satisfy the threshold criteria
5. Computationally validate and predict the associations of these genes using our in-house protein interaction algorithm.
6. Extract the Gene Characterization Index score (GCI) for the genes in the dataset.
7. Extract the Gencards Inferred Functionality score (GIfTs) score for the genes in the dataset.
8. Using Bayes server 3.0, construct static Bayesian networks on the basis of interactions and the topological information from KEGG database and Metacore, at 12 and 24-hours respectively.
9. Train the model for a tentative number of iterations until the desired prediction levels have been met.
10. Simulate the 12-hour static network to 24 and 36-hours dynamic Bayesian network and 24-hour into 36-hour respectively. This is called forward dynamic simulation of Bayesian network.

11. Observe the topological changes at different time points in the DBNs.
12. Note down the node probability distributions and plot them using line plots.

1.3 Contribution of the Thesis

The proposed methodology is to assess the prediction performances of DBNs constructed and simulated using time-course breast cancer gene expression data, when sufficient knowledge parameters are supplied. A Bayesian model has been developed to incorporate parameters that are time-point independent, whose inclusion in the node probability distributions explains the importance of scores derived from literature annotations, thereby adding a weight to the identified targets. This model considers the time-course gene expression data along with the ability to work with non-time-course gene expression data. A comparison study between the conditional probability distributions (CPDs) at the nodes in the static bayesian network stage to the nodes at dynamic stage would give an insight into the model's prediction performance. Statistical analysis are carried out to explain the node CPD changes and to validate the comparison studies.

1.4 Organization of the Thesis

The rest of the thesis is organized as follows. Chapter 2 describes previous work done in the area of DBNs, Bayesian information criterion, gene expression data simulations, node and edge properties in a DBN and validation techniques employed in the DBN simulation studies. Chapter 3 explains the design of methodology and implementation details, where it describes the type of simulations to be performed to prove the hypothesis. Chapter 4 outlines the Bayesian model training, topological

changes in the model, different time-point and data combinations with test datasets and experimental dataset. It also reviews about identifying the node CPDs after simulations and the validation techniques that have been employed. Finally, Chapter 5 discusses the results and outcomes of this analysis. Chapter 6 concludes on limitations in this study and future work to be performed to implement the model more efficiently.

2. LITERATURE REVIEW

For centuries, modeling biological systems has been a researcher's principal tool for understanding the complex mechanisms. Models exist in all disciplines ranging from *A*strophysics to *Z*oology. Numerical models dominate the modeling practice, for dynamic models the usage of differential equations was dominant. A Bayesian model is a simplified picture of a part of the real world that explains and lets understand the reality. *Newton's three laws of motion* acts as a very good model due to its' simplicity, reliability and predictive power. The reason is that Newton's laws prescribe exactly what parameters we need, such as the gravitational constant, air resistance, weight of the tile, and height of the building. Furthermore, it prescribes the exact relations between the variables. In this case, it is relatively easy to make a good model, because we know the underlying relations and we can measure all parameters directly. Microarray experiments even though are on the verge of extinct in the light of next-generation sequencing experiments, have contributed a lot to the life sciences industry with information on gene expression and sequences. Considering the complex system of physiological processes in a living human being that are responsible for a gene's activity, it is an exhaustive task of simulating the same process in real time. The introduction of computers in the biological domain, and the use of ordinary differential equations have made it easier to understand the passage of information among the genes. Many experiments have been carried out on simulating gene expression data using Bayesian methodologies. This research throws light to assess the prediction performance of such simulations in a dynamic environment when prior knowledge parameters are incorporated into the Bayesian model.

2.1 Bayesian Networks

A Bayesian network (BN), also known as a belief network is a graphical model for probabilistic relationships among a set of variables (Heckerman, 1998). Over the years, expert systems use BNs in domains where uncertainty plays an important role. Nowadays, BNs are used in a wide range of diagnostic medical systems, fraud detection systems, missile detection systems, etc.

BNs have a couple of properties that make them so popular and suitable to use. The five most important properties are, in no particular order, the following: (1) Ability to handle incomplete data sets; (2) Enabling learning about relationships between variables; (3) Combining expert knowledge and data into a BN; (4) Use of Bayesian methods bypasses the over fitting of data during learning and can be avoided relatively easy; and (5) Ability to model causal relationships between variables.

There are two components in a BN - qualitative and quantitative. The qualitative component is represented by the topology of the network and the quantitative component is expressed by the assignment of the conditional probability distributions to the nodes. Before getting into the detailed descriptions of the above mentioned components, Bayes' Theorem is presented below.

$$p(X|Y) = \frac{p(Y|X) \cdot p(X)}{p(Y)}$$

where

• p (X|Y) is the posterior probability of the hypothesis X, given the data Y,

• p (Y|X) is the probability of the data Y, given the hypothesis X, or the likelihood of the data,

• p (X) is the prior probability of the hypothesis X, and

• p (Y) is the prior probability of the data Y, or the evidence.

2.1.1 Bayesian Network Structure

The qualitative part of a BN is represented by its structure. A BN is a directed, acyclic graph (DAG) where the nodes represent the set of random variables and the directed arcs also called as edges connect the nodes. The nodes in the BN represent *discrete* random variables. If an arc points from *X*1 to *X*2 then *X*1 is a *parent* of *X*2 and *X*2 is a *child* of *X*1. A parent directly influences its children. Furthermore, every node in the BN has its own local probability distribution. All these components together form the joint probability distribution (JPD) of the BN.

The process of breaking up the joint probability distribution into local probability distributions is called *factorization*, which results in an efficient representation that supports fast inference. This is a property of BNs that forms a major contribution to its success.

2.1.2 Conditional Probabilities

The quantitative part of the BN is represented by the assignment of the conditional probability distributions to the nodes. Each node in a BN has a *conditional probability table* (CPT) that defines the conditional probability distribution (CPD) of the represented discrete random variable. CPTs tell us what the probabilities are of a hidden node given its parents. The size of the CPTs grows exponentially with the number of

parents. Knowing this, it is important to keep the number of nodes that node *Xi* is dependent on as low as possible.

2.1.3 Inference

Inference is defined as the process of calculating the probability of one or more variables ***X,*** given some evidence ***e***. The evidence is expressed as an instantiation of variables in the BN. In short: $p(\boldsymbol{X}|\boldsymbol{e})$ needs to be calculated. The two rules that are needed for inference are Bayes' theorem and the expansion rule. Exact inference of large, multiply connected BNs becomes very complex. In fact, it is *NP-hard* (Cooper, 1990). Because of this intractability, approximate inference methods are essential. In general, r*andomized sampling algorithms*, also called *Monte Carlo algorithms*, are used. For exact inference, *variable elimination* algorithms or *junction tree* algorithms can be used. Variable elimination uses a smart way to rewrite the inference calculations by choosing a specific elimination order of the variables, making computation more efficient. Junction tree algorithms convert the multiply connected BNs to *junction trees*, after which inference can be performed using variable elimination (Shafer & Shenoy, 1988) or belief propagation (Lauritzen & Spiegelhalter, 1988). Of course, this conversion can be very complex as well, so it is not suitable for every BN. A third technique that can be used to reduce complexity of the inference computations is called *relevance reasoning*. This is a preprocessing step that explores structural and numerical properties of the model to determine what parts of the BN are needed to perform the computation. Relevance reasoning reduces the size and the connectivity of a BN by pruning nodes that are computationally irrelevant to the nodes of interest in the BN (Druzdzel & Suermont, 1994).

2.1.4 Conditional Independence

Conditional independence can cut the cost of inference calculations drastically, turning the complexity from 2^n to $n \cdot 2^k$ were n is the total number of variables and k the maximum number of variables that a single variable can be dependent on. Variables with no connections at all are conditionally independent, but variables with an indirect connection can still be conditionally dependent. The variables in a BN are conditionally independent if and only if they are d-separated.

2.1.5 D-separation

Two nodes X1 and X3 in a BN are d-separated if for all paths between X1 and X3, there is an intermediate node X2 for which either:

• When the state of X2 is known, the connection is serial or diverging; or

• Neither X2 nor any of its descendants have received any evidence at all and the connection is converging.

For large BNs, d-separation of two variables can be solved using the Bayes ball algorithm, a reachability algorithm that makes use of the notion of a ball and a set of bouncing rules to determine whether a set of variables can be reached from another set of variables through a third set of variables (Shachter, 1998).

2.1.6 Learning

One very useful property of BNs is their ability to learn from observations. Learning of BNs can be divided into two types: *parameter learning* and *structure learning*. With parameter learning, the structure of the BN is given and only the CPT

parameters are learned. With structure learning, the BN structure itself is learned. Before parameter learning and structure learning are discussed, a short introduction on the concept of *Bayesian learning* is mentioned in the next section.

2.1.6.1 Bayesian learning

Bayesian learning calculates the probability of each of the hypotheses given the data. Predictions are made by averaging over all probabilities of the hypotheses. Real-life problems have an enormous hypothesis space and it becomes a complex process of learning. A common approximation is *Maximum A Posteriori* (MAP) estimation. Instead of all hypotheses, only the best hypothesis, which is the *hi* that maximizes $p(hi|\mathbf{d})$, is taken: $p(X|\mathbf{d}) \approx p(X|h\text{MAP})$. Another simplification is assuming an equal prior for all hypotheses, as is the case in the biased coin example. In this case, MAP estimation reduces to choosing the *hi* that maximizes $p(\mathbf{d}|hi)$. This is called *maximum likelihood* (ML) estimation and provides a good approximation as long as the data set is large enough.

Structure/Observability	Method
Known and full	Sample Statistics
Known and partial	EM or gradient ascent
Unknown and full	Search through model space
Unknown and partial	Structural EM

Table 1.1: Methods adapted for different combination of Bayesian network topological knowledge

Full and partial observability indicates that the values of all variables are known and do not know the values of some of the variables, respectively. This might be because they cannot be measured in principle, or because they just happen to be unmeasured in the training data. Note that a variable can be observed intermittently. Unknown structure means we do not know the complete topology of the graph except some parts or properties it is likely to have, e.g., it is common to assume a bound, on the maximum number of parents that a node can take on, and we may know that nodes of a certain "type" only connect to other nodes of the same type. Such prior knowledge can either be a "hard" constraint or a "soft" prior. Best use of such knowledge is possible only by using Bayesian methods, which have the additional advantage that they return a distribution over possible models instead of a single best model. Handling priors on model structures is quite complicated and shall not be discussed (Heckerman, 1998). However, the assumption is that the goal is to find a single model, which maximizes scoring function. When DBNs are considered, in which all the variables are continuous, numerical priors are used as a proxy for structural priors. An interesting approximation to Bayesian approach is to learn a mixture of models, each of which has different structure, depending on the value of hidden or discrete or mixture nodes. This has been done for Gaussian networks and discrete, undirected trees; unfortunately, there are severe computational difficulties in generalizing this technique to arbitrary, discrete networks.

2.1.6.2 Parameter learning

A BN consisting of discrete random variables has a CPT for every variable. Every entry Φ in a CPT is unknown at first and must be learned. Let's take the coin toss

example. Suppose nothing is known about how the coin is biased. This means that it can be biased toward heads with a probability $0 < \Phi < 1$ and toward tails with a probability $0 < (1 - \Phi) < 1$.

This means that it can be biased toward *heads* with a probability $0 < \Phi < 1$ and toward *tails* with a probability $0 < (1 - \Phi) < 1$. Using ML estimation, only an expression for $p(\boldsymbol{d}|\Phi)$. and its maximum needs to be found to obtain the optimal value for the parameter Φ. In the coin example, this will result in a hypothesis equal to the proportion of tosses that gave *heads* divided by the total number of tosses. The obvious problem with this approach is the large data set needed to get a satisfactory result. Such a large data set is not available most of the time. The solution to this problem is introducing a hypothesis prior to the possible values of the needed parameter. This prior is updated after each new observation. In the coin example, the parameter Φ is unknown. Instead of giving it one prior value, a continuous prior distribution $p(\Theta)$ is assigned to it. A *beta distribution* is used to achieve this. In parameter learning, usually parameter independence is assumed, meaning that every parameter can have its own beta or Dirichlet distribution that is updated separately as data are observed.

The assumption of parameter independence results in the possibility to incorporate the parameters into a larger BN structure. The process of learning BN parameters can be formulated as an inference problem of an extended BN, making it possible to use the same techniques as discussed in inference to solve the learning problem $p(\Phi|\boldsymbol{d})$. In the case that the data is incomplete or partially observed, approximation algorithms like EM algorithm needs to be used.

2.1.6.3 Structure learning

Next to the parameters of the BN, also its structure can be learned from a data set. This is called *structure learning*. There are two types of structure learning: *constraint-based learning* and *score-based learning*. Constraint-based learning tries to find the optimal network structure based on a set of independence constraints. These independence constraints can be learned from data, but that is a statistically difficult task. In score-based learning, a scoring function is defined that consists of one or more search criteria that assign a value to each network structure. A searching algorithm, such as hill-climbing or simulated annealing (Cheng J et al, 2002), is used to find the maximum value of this scoring function. A network structure does not have to be learned from ground up, but can be derived from expert knowledge. After that, a structure learning technique can be used to optimize the derived network structure. Structure learning is not in the scope of this thesis and interested readers are directed to (Heckerman, 1998, & Koller, Friedman, 2005).

2.2 Dynamic Bayesian Networks

A BN is useful for such domains where the state of parameters is static. In such condition, every variable has a single and fixed value. Unfortunately, this assumption of static condition is not always sufficient, as we have seen with the Navy BN in the introduction of this chapter. A *dynamic Bayesian network* (DBN), which is a BN extended with a time dimension, can be used to model dynamic systems (Dean, Kanazawa, 1988). In this section we only describe the DBN formalism that is in common use today. Our extension of the formalism and its application is described in parts II and

III of this thesis. A more extensive introduction on DBNs can be found in: (Jordan, 2003, & Murphy, 2002).

2.2.1 Dynamic Bayesian network structure

First of all, the dynamic extension does not indicate that there will be a dynamic change in the network structure or parameters, but that a dynamic system is modeled. A DBN is a directed acyclic graphical model of a stochastic process. It consists of time-slices with each slice containing its own variables. A DBN is defined as the pair ($B1$, B) where $B1$ is a Bayesian network that defines the prior or initial state distribution of the state variables p ($\boldsymbol{Z}1$) (Murphy, 2002) and B is a two-slice temporal Bayesian network (2TBN) that defines the transition model.

2.2.2 Inference

There exist several ways of performing inference on DBNs. The most common types of inference are given in figure 3.12.

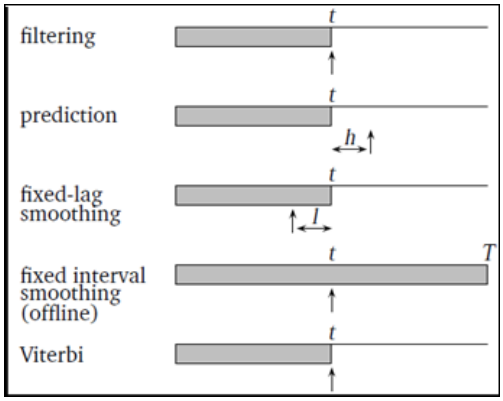

Figure 2.1: Different types of inference. Arrow indicates the time of reference. t represents current time and T, the sequence length. (Hulst J, 2006)

- With *filtering* (or *monitoring*), the current belief state is computed given all evidence from the past. To achieve this, $p(\boldsymbol{X}t|\boldsymbol{y}1{:}t)$ needs to be calculated. Filtering is used to keep track of the current state for making rational decisions. It is called *filtering*, because noise is filtered out from the observations.

- With *prediction*, a future belief state is computed given all evidence from the past. This means $p(\boldsymbol{X}t+h|\boldsymbol{y}1{:}t)$ needs to be calculated for some $h > 0$. Prediction can be used to evaluate the effect of possible actions on the future state.

- With *smoothing* (or *hindsight*), a belief state in the past is calculated when all the evidences up to the present are given. So, $p(\boldsymbol{X}t-l|\boldsymbol{y}1{:}t)$ is calculated for some fixed time-lag $l > 0$. Smoothing is useful to get a better estimate of the past state, because more evidence is available at time t than at time $t - l$. In figure 3.12, two types of smoothing are given: *fixed-lag smoothing*, which is the type of smoothing given above, and *fixed-interval smoothing*, which is the offline case in which $p(\boldsymbol{X}t|\boldsymbol{y}1{:}T)$ is calculated for all $1 < t < T$.

- The final inference method to be described here is *Viterbi decoding* (or *most likely explanation*). This is a different kind of inference, but it is used very often nonetheless. With Viterbi decoding, given a sequence of observations, one might compute the most likely sequence of hidden states. That is: $\arg\max \boldsymbol{x}1{:}t\ p(\boldsymbol{x}1{:}t|\boldsymbol{y}1{:}t)$ needs to be calculated.

In general, there are two types of inferences. They are exact inference and approximate inference.

2.2.2.1 Exact inference

The first approach for exact inference in DBNs is based on the notion that an unrolled DBN is in fact the same as a static BN. *Variable elimination* can be used with filtering. This algorithm results in having only two slices of the unrolled DBN in memory at a time. For smoothing, using the *junction tree algorithm* is more efficient. A second approach is to convert the DBN to a HMM and then apply the *forward-backward algorithm*. Converting a DBN to a HMM is only possible with discrete state DBNs. With *Nh* number of hidden variables per slice, each variable having up to *M* values, the resulting HMM will have maximally $S = MNh$ values. As long as *S* is not too large, this is a nice method, because it is very easy to implement. Unfortunately, most of the times *S* will be too large and more efficient methods need to be used.

The *frontier algorithm* (Zweig, 1996) is based on the notion that in the forward-backward algorithm, *Xt* of the HMM d-separates the past from the future which can be generalized to a DBN by noticing that the hidden nodes in a slice d-separate past from future. This set of nodes is called *the frontier*. This algorithm also uses a forward and a backward pass. The *interface algorithm* (Murphy, 2002) is an optimization of the frontier algorithm because it does not use all hidden nodes in a slice, but only the subset of nodes with outgoing arcs to d-separate the past from the future. This subset is called the forward *interface*. The modified 2TBN is called a *1.5DBN Ht* in this algorithm, because it contains all nodes from slice 2, and nodes with an outgoing arc from slice 1. After this modification, a junction tree is created of each *Ht* and they are *glued* together. Finally,

inference is performed on each separate tree and messages are passed between them via the interface nodes, first forward and then backward. Finally, analogous to the conversion from discrete state-space DBN to HMMs, a linear- Gaussian state-space DBN can be converted to a Kalman filter model (KFM). *Kalman filtering* or *Kalman smoothing* can be used for exact inference after this conversion.

2.2.2.2 Approximate inference

Exact inference in discrete-state models is often unacceptably slow. When faster results are needed, approximate inference can be used. Furthermore, it turns out that for most DBNs with continuous or mixed discrete-continuous hidden nodes, exact exhibition of the belief state do not exist. Generally, two types of approximations are distinguished: *deterministic* and *stochastic* algorithms. Deterministic algorithms for the discrete-state DBNs include: the *Boyen-Koller algorithm* (BK) (Boyer, Koller, 1998) where the interface distribution is approximated as a product of marginals and the marginals are exactly updated using the junction tree algorithm; the *factored frontier algorithm* (FF) (Murphy, Weiss, 2001), where the frontier distribution is approximated as a product of marginals and the marginals are computed directly; and *loopy belief propagation*, which is a generalization of BK and FF. Deterministic algorithms for linear-Gaussian DBNs include: the *Generalized Pseudo Bayesian* approximation (GPB(n)), where the belief state is always represented using $Kn-1$ Gaussians; and the *interacting multiple models* (IMM) approximation, which also collapses the priors. Deterministic algorithms for mixed discrete-continuous DBNs include: *Viterbi approximation*, in which the discrete values are enumerated in a priori order of probability; *expectation propagation*, which is the generalization of BK for discrete-state DBNs and GPB for linear Gaussian DBNs;

and *variational methods*, that decompose the problem into separate discrete and continuous chains which are treated differently.

Deterministic algorithms for non-linear or non-Gaussian models include: the *extended Kalman filter* (EKF) or the *unscented Kalman filter* (UKF) for non-linear models, which calculate a local linearization and then apply a Kalman filter (KF). Stochastic algorithms can be divided into offline and online methods. Offline methods are often based on *Monte Carlo Markov Chain (MCMC) or importance sampling*. Online methods often rely on methods such as *particle filtering*, *sequential Monte Carlo*, the *bootstrap filter*, the *condensation algorithm*, *survival of the fittest*, and others. Stochastic algorithms have the advantage over deterministic algorithms that they are easier to implement and are able to handle arbitrary models. Unfortunately, this comes with a price, because stochastic algorithms are generally slower than the deterministic methods. Both methods can be combined to get the better of two worlds. (Murphy, 2002) presents the *Rao- Blackwellised Particle Filtering algorithm*, in which some of the variables are integrated out using exact inference and sampling is applied to the other variables.

2.2.3 Learning

The techniques for learning DBNs are generally the same as the techniques for learning static BNs. The specific methodology used in this research for learning the parameters of a DBN with complete data is presented below. Current DBN parameter learning algorithms for Murphy's formalism unroll a DBN for $t = T$ time-slices, where T is the sequence length of the process, after which the learning data is inserted in the unrolled network and the parameters are learned. Remember that for Murphy's

formalism, two sets of parameters need to be specified, the initial parameters in $B1$, a BN that defines the initial state distribution of the state of variables and the transition parameters in the second slice of B , a two-slice temporal BN that defines the transition model. An example for learning a DBN model of a HMM is shown schematically in figure 2.2.

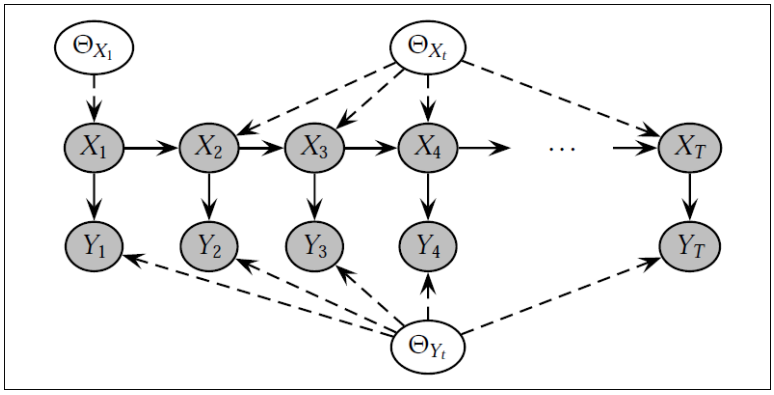

Figure 2.2: Unfolding DBN during its' parameter learning stage with no connection between the initial and simulated parameter. (Hulst J, 2006)

In this example, the parameters for $B1$ are given by $\{\Theta X1,\Theta Yt\}$ and the parameters for B are given by $\{\Theta Xt\ ,\Theta Yt\}$ When learning the parameters, the DBN is unrolled for T time-slices, the parameters are tied and learned by insertion of the learning data. However, in case the initial parameters represent the stationary distribution of the system, they become coupled with the transition parameters. This is shown schematically in figure 2.3.

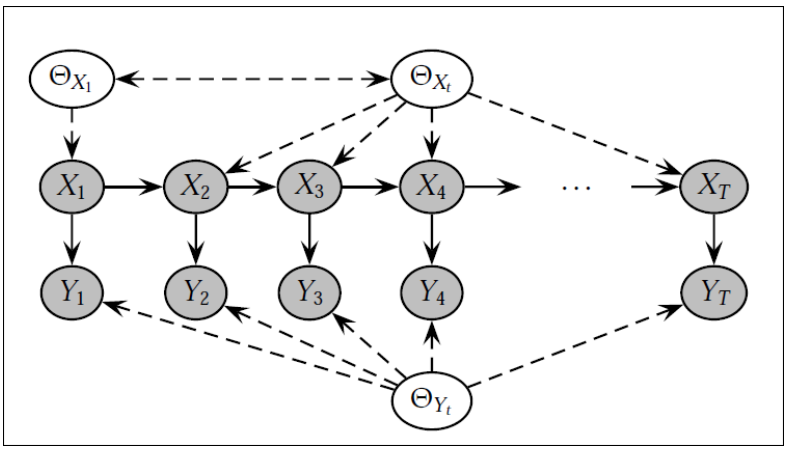

Figure 2.3: Unfolding DBN during its' parameter learning stage with connection between the initial and simulated parameter. (Hulst J, 2006)

On a modeling level, decoupled initial and transition parameters mean that the modeled process has a specific beginning, i.e. the first time-slice of the DBN always models the same time instant of the process. The initial and transition parameters become coupled when the modeled process does not have a specific beginning, i.e. the first time-slice of the DBN starts somewhere in the modeled process, but where is unclear. When modeling physiological processes, we are dealing with the latter, because a patient does not have a distinctive initial state. As we mentioned already in the previous section, learning DBN parameters in case the initial parameters represent the stationary distribution of the system can be problematic. For first-order DBNs, Nikovski presents an algorithm, but this algorithm cannot be used with the DBN formalism.

The main differences between learning of static and DBNs are parameter learning and structural learning.

Parameter learning can be done both online and offline. For online learning, the parameters are added to the state space after which online inference is applied. For offline

learning, the same techniques as for learning BNs can be used. Some points that are specifically applicable to DBNs:

- Parameters must be joined to time-slices, so models of unbounded length can be modeled.
- In the case that the initial parameters π represents the stationary distribution of the system, they become coupled with the transition model. As such, they cannot be estimated independently of the transition matrix (Nikovski, 1998).
- Linear-Gaussian DBNs sometimes have closed-form solutions to the maximum likelihood estimates.

Structural learning of DBNs consists of learning both inter-slice and intra-slice connections. If only the inter-slice connections are learned, the problem reduces to feature selection (what variables are important for temporal reasoning and what variables are not). Again, learning for static BNs can be adapted.

2.3 Other temporal reasoning techniques

The DBN formalism is not the first development in temporal reasoning under uncertainty. The two most popular techniques still in use nowadays are the hidden Markov model (HMM) and the Kalman filter model (KFM). Their popularity is mostly due to their compact representation, fast learning and fast inference techniques. However, a DBN can have some significant advantages over these two formalisms. For one, Hidden Markov models and Kalman Filter models are really limited in their expressive power. In fact, it is not even correct to call HMMs and KFMs other techniques, because the DBN formalism can be seen as a generalization of both HMMs and KFMs. The DBN

formalism brings out connections between these two models that were assumed to be different, previously, because they were developed in very different research areas. DBNs generalize Hidden Markov Models by allowing the state space to be represented in factored form, instead as a single discrete random variable. DBNs generalize Kalman Filter Models by allowing arbitrary probability distributions, not just conditional linear Gaussian distributions. The basic versions HMMs and KFMs are briefly discussed here because of historical relevance. Extensions exist for both HMMs and KFMs to counter some of their issues, but an extensive description of those is beyond the scope of this thesis.

2.3.1 Hidden Markov Models

A HMM models a first-order Markov process where the observation state is a probabilistic function of an underlying stochastic process that produces the sequence of observations. The underlying stochastic process cannot be observed directly, it is *hidden*. Both the hidden and observation states are modeled by discrete random variables [Rab89]. The HMM formalism first appeared in several statistical papers in the mid-1960s, but it took more than 10 years before its usefulness was recognized. Initially, the use of HMMs was a great success, especially in the fields of automatic speech recognition (ASR) and bio-sequence analysis. Because of its success, the use of HMMs in ASR is still dominant nowadays, despite its undeniable issues (Russel, Blimes, 2003).

A HMM definition consists of three parts: an initial state distribution π_0, an observation model $p\,(Yt|Xt)$ and a transition model $p(Xt|Xt-1)$. HMMs are generally used to solve one of the following problems: (1) state estimation $p\,(Xt = x|\mathbf{y}1{:}t)$ using the *forward-backward* algorithm, (2) most likely explanation arg max$\mathbf{x}1{:}t\ p(\mathbf{x}1{:}t|\mathbf{y}1{:}t)$ using

the *Viterbi* algorithm, and/or (3) maximum likelihood HMM arg maxλ $p(\boldsymbol{y}1{:}t|\lambda)$ (for learning the HMM) using the *EM* algorithm, called the *Baum-Welch* algorithm in this context.

One main problem among many of HMMs is the fact that the hidden state is indicated by only a single discrete random variable. DBNs are able to decompose the state of a complex system into its constituent variables, taking advantage of the sparseness in the temporal probability model. This can result in exponentially fewer parameters. The effect is that using a DBN can lead to fewer space requirements for the model, less expensive inference and easier learning.

2.3.2 Kalman filter models

Basically, a KFM is a HMM with conditional linear Gaussian distributions (Welcha, Bishop, 2004). It is generally used to solve uncertainty in linear dynamic systems. The KFM formalism first appeared in papers in the 1960s (Kalman, 1960), and was successfully used for the first time in NASA's Apollo program. Nowadays, it is still used in a wide range of applications. The KFM formalism assumes that the dynamic system is jointly Gaussian in nature which means the belief state to be unimodal that is inappropriate for many problems. The main advantage of using a DBN over a KFM is that the DBN can use arbitrary probability distributions instead of a single multivariate Gaussian distribution.

2.3.3 Dynamic Conditional Random Fields

Conditional random fields (CRFs) is a undirected graphical probabilistic framework to label and segment structured data on the idea of defining a conditional probability distribution given a particular observation, rather than a joint distribution over both labeled and observed information. CRFs have an advantage over HMMs in terms of their conditional nature, resulting in the relaxation of the independence assumptions required by HMMs in order to ensure tractable inference. CRFs avoid the label bias problem, outperforms both maximum entropy Markov models (MEMM) and HMMs on a number of real-world tasks in the fields of bioinformatics, computational linguistics and speech recognition (Charles S, et al, 2007).

Dynamic conditional random fields (DCRFs) are a generalization of linear-chain conditional random fields (CRFs) in which each time slice contains a set of state variables and edges. They represent distributed states as in dynamic Bayesian networks (DBNs), and parameters are tied across slices. DCRFs allows to represent distributed hidden state and complex interaction among labels, as in DBNs, and to use rich, overlapping feature sets, as in conditional models. DCRF performs better than CRFs, achieving maximum performance using only half the training data. In addition to maximum conditional likelihood, there are two alternative approaches for training DCRFs: marginal likelihood training and cascaded training. Among the two, marginal training can improve accuracy when uncertainty exists over the latent variables. Any DCRF with multiple state variables can be collapsed into a linear-chain CRF. However, such a linear chain CRF needs exponentially many parameters in the number of variables.

Like DBNs, DCRFs represent the joint distribution with fewer parameters by exploiting conditional independence relations.

2.4 Softwares for performing Bayesian simulations

The software with DBN functionality that is currently on the market can be divided into two classes: software that does have a GUI and software that does not have a GUI. The software that does have a GUI is mostly commercial and is relatively easy to use for a user with only an average knowledge of BNs. The software without a GUI is mostly academic-oriented, which means that it is a result of research in the area of DBNs. Academic-oriented software is very flexible, but this can also make its application to specific problems very time-consuming.

The Inference, Structural Modeling and Learning Engine are fully platform independent libraries written in C++ classes that implements graphical probabilistic and decision-theoretic models that are suitable for direct inclusion in intelligent systems (Druzdzel, 1999). The individual classes defined in the Application Program Interface (API) of SMILE enable the user to create, edit, save and load graphical models, and use them for probabilistic reasoning and decision making under uncertainty. To be able to access the SMILE library from a number of other programming languages, wrappers exist for ActiveX, Java, and .NET. SMILE was first released in 1997.

The Graphical Network Interface is the graphical user interface to the SMILE library. It is implemented in C++ and makes heavy use of the Microsoft Foundation Classes. Its emphasis is on accessibility and friendliness of the user interface, making creation of decision theoretic models intuitive using a graphical click-and-drop interface

approach which is user-friendly and versatile environment for building graphical decision models and performing diagnosis.

Three libraries that have DBN functionality and are freely downloadable include the Bayes net toolbox for MATLAB, graphical models toolkit (GMTK), probabilistic network library (PNL).

2.4.1 Bayes net toolbox for MATLAB

Until the introduction of the Bayes net toolbox for Matlab (BNT) (Murphy, 2001), the field lacked a free general-purpose software library that was able to handle many different variants of graphical models, inference and learning techniques. The BNT is an attempt to build such a free, open-source, and easy-to-extend library that can be used for research purposes. The author chose to implement the library in Matlab because of its ease of handling Gaussian random variables. Matlab has various advantages and disadvantages, the main disadvantage being that it is terribly slow.

In the BNT, BNs are represented as a structure containing the graph, the CPDs and a few other pieces of information. The BNT offers a variety of inference algorithms, each of which makes different tradeoffs between generality, accuracy, speed, simplicity, etc. All inference methods have the same API, so they can be easily interchanged. The conditional probabilities of the defined variables can be continuous or discrete. Parameter and structure learning are supported as well.

The toolbox was the first of its kind and set a standard for DBN libraries. It is still widely in use today, because the code is relatively easy to extend and well documented and the library has by far the most functionality of all software currently available.

However, much functionality still needs to be added to make it a real general-purpose tool, such as tree-structured CPDs, Bayesian modeling, online inference and learning, prediction, more approximate inference algorithms, and support for non-DAG graphical models. Also, the scripting part that is needed to implement a model in the BNT can be really cumbersome. Added to that, BNT does not have support for standard BN file formats, which cancels the possibility to export and/or import BN models from/to other packages. The BNT is released as open-source software and can be downloaded from http://bnt.sourceforge.net. A GUI is currently in development.

2.4.2 Graphical Model Toolkit

The graphical models toolkit (GMTK) (Bilmes, Zweig, 2002) is a freely available and open-source toolkit written in C++ that is specialized in developing DBN-based automatic speech recognition (ASR) systems. The GMTK has a number of features that can be used for a large set of statistical models. These features include several inference techniques, continuous observation variables and discrete dependency specifications between discrete variables. The DBN model needs to be specified in a special purpose language. In this language, a DBN is specified as a template that contains several time-slices. The collection of time-slices is unrolled over time to create an unrolled DBN. The time-slices in the template are divided into a set of prologue, repeating and epilogue time-slices and only the repeating frames are copied over time. This approach to modeling DBNs has much expressive power, but it is also a lot of work to specify a DBN model. The GMTK is a promising library. To become really useful, a couple of disadvantages need to be tackled. Its main disadvantages are that it is not a general-purpose toolkit,

because it specializes in ASR and it therefore lacks much functionality that is useful for other applications. Furthermore, the documentation is far from complete, making it difficult for a user that is not closely involved in the development of the software to understand the toolkit. Finally, although the author of the toolkit states that it is open-source, the website (http://ssli.ee.washington.edu/~bilmes/gmtk) still only offers a binary version.

2.4.3 Probabilistic network library

Intel's research lab in Saint Petersburg, Russia is responsible for the probabilistic network library (PNL) (Intel Corporation, 2004), which is an open-source library written in C++ that can be used for inference and learning using graphical models. PNL has support for a large variety of graphical models, including DBNs. In fact, PNL can be seen as the C++ implementation of BNT, since its design is closely based on that library. PNL does not support all functionality provided by BNT yet, but the long-term goal is that it will surpass that functionality. The influence of BNT is very clear when diving into the API of PNL. Because the approach is similar, PNL can also only model first-order processes. The API is very well documented; making it a good option if one wants to model DBNs. However, the implementation is far from complete and still lacks much functionality that BNT offers. The library recently reached its v1.0 status, which can be downloaded from http://www.intel.com/technology/computing/pnl. Work is being done to make PNL compatible with the GeNIe GUI.

2.4.4 DBN modeling tools

Currently, there are three softwares that support temporal reasoning and have a GUI; they are BayesServer, BayesiaLab and Netica. A detailed introduction to these softwares is presented below.

2.4.4.1 BayesServer

BayesServer is a tool for modeling Bayesian networks and Dynamic Bayesian networks developed by the BayesServer Corporation from United Kingdom. Bayes Server 3.0 supports both Bayesian networks and Dynamic Bayesian networks, especially in time series analysis, and has a rich user interface and API for building and visualizing models, learning models from data, sampling data, charting, and building complex probability queries, including time series predictions. The tool supports end-users with a forum to discuss the current trends, any glitches in the process of simulations and updates in new releases.

The flowchart of simulating time series Bayesian networks using BayesServer usually follows a series of steps. The user creates the temporal nodes using the network tab and assigning the links between the nodes (Figure 2.4). Parameter learning in BayesServer can be done by loading the distribution parameters manually using the distribution editor or by making the model learn automatically from the data (Figure 2.5). BayesServer also supports data plots to view the distribution of data (Figure 2.7). After the learning phase is finished, data connections using the data explorer (Figure 2.6) are made. Data connections are nothing but mapping the data columns in the data set to the

respective nodes in the Bayesian network followed by entering the evidence. Following the data-mapping phase, the candidate network alias the static Bayesian network is visualized in the BayesServer GUIs. Some of the screenshots related to the above mentioned phases have been presented.

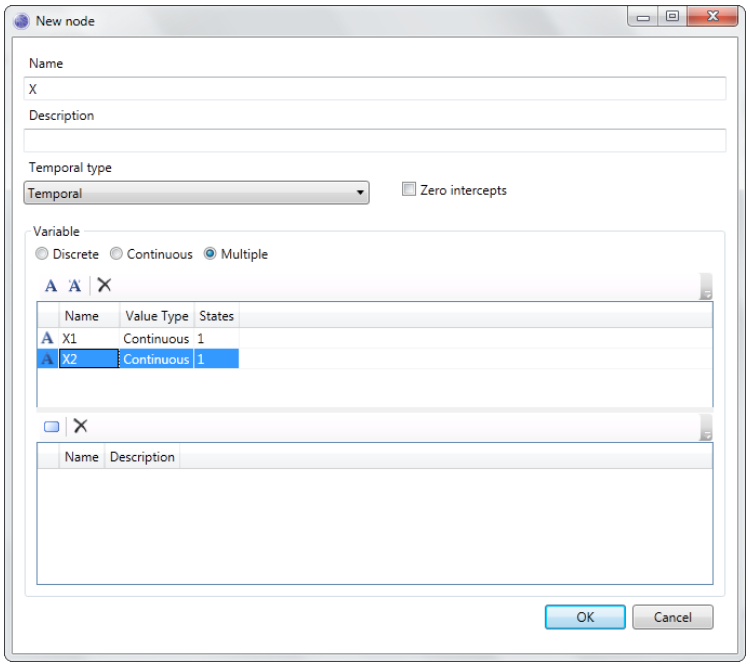

Figure 2.4: Creating a node in BayesServer

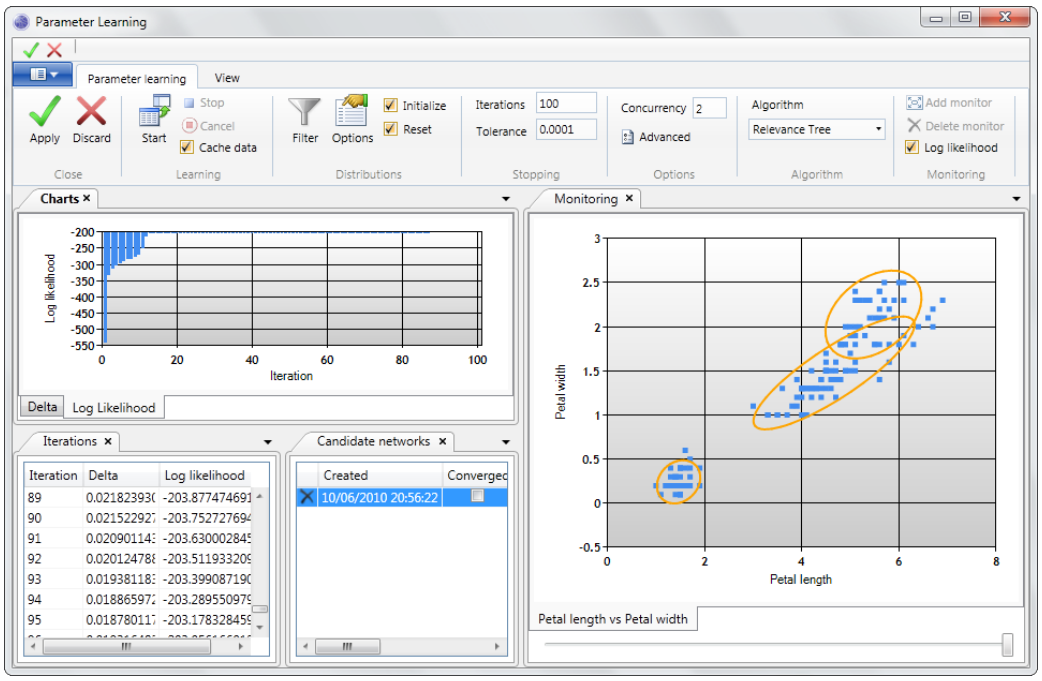

Figure 2.5: Learning Parameters in BayesServer

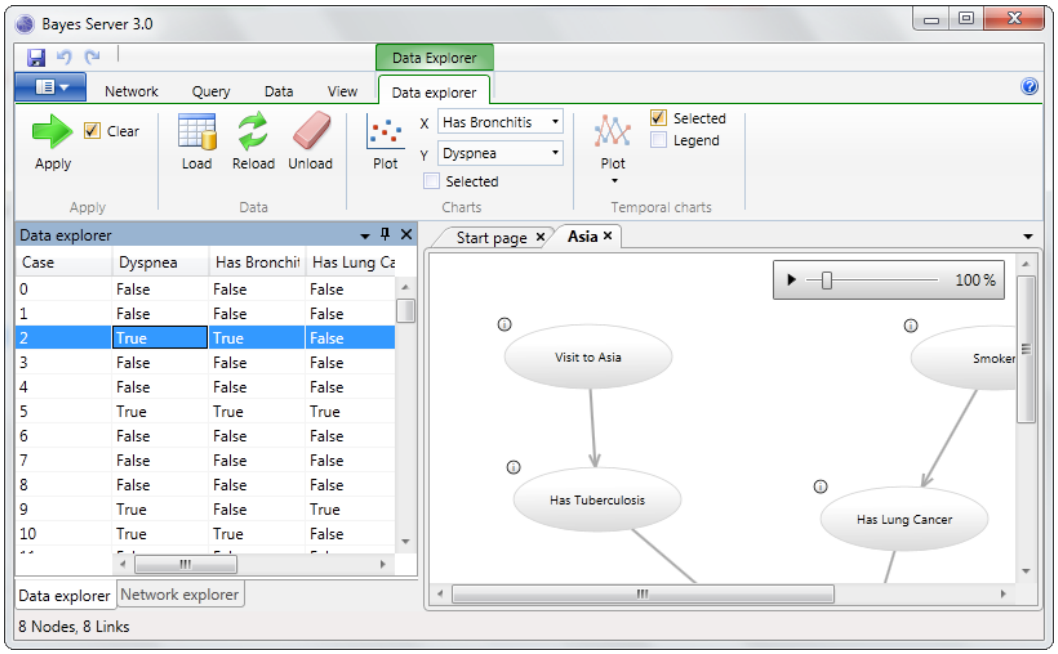

Figure 2.6: Data explorer section of BayesServer

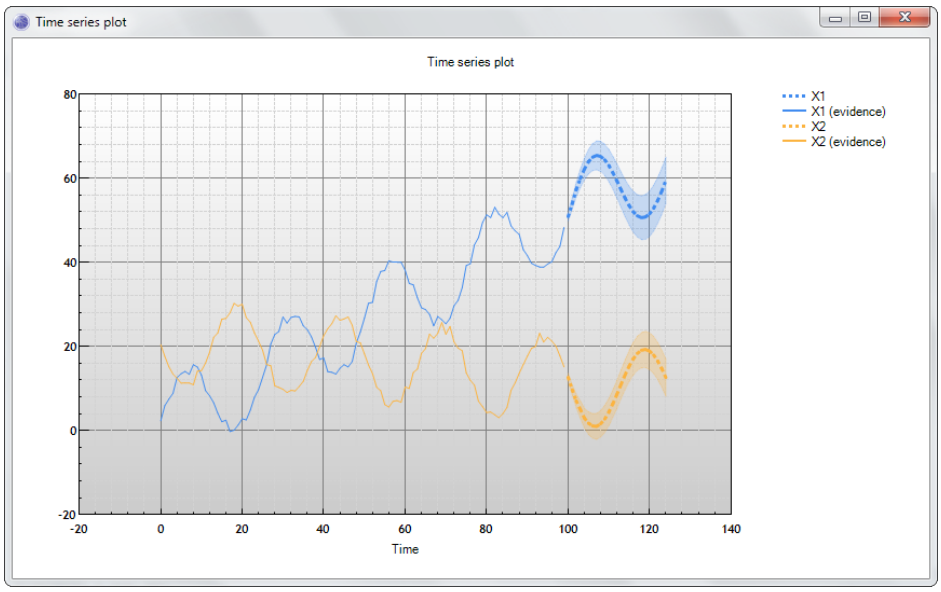

Figure 2.7 (a): Time series data plot in BayesServer

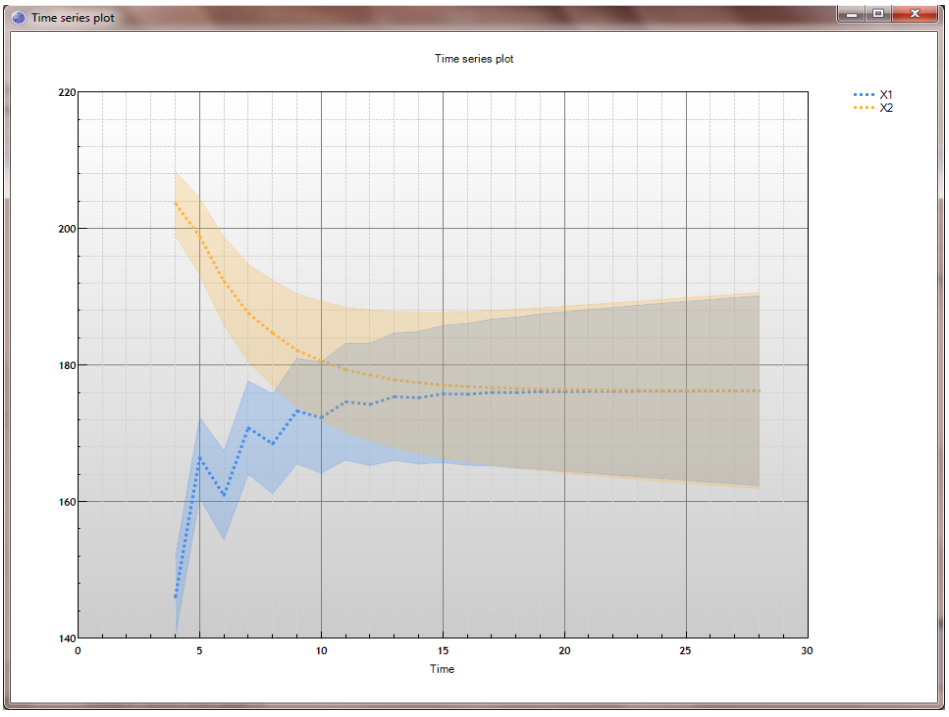

Figure 2.7 (b): Time series data plot in BayesServer

2.4.4.2 BayesiaLab

BayesiaLab is the BN software toolkit developed by the French company Bayesia (http://www.bayesia.com). BayesiaLAB has two modes: a *modeling mode* for designing the BN and a *validation mode* for inference. In the modeling mode, it is possible to add temporal arcs to a BN to indicate that the parent node of the two nodes is in the previous time-slice. Only first-order Markov models can be designed this way. Temporal arcs have a different color. The user needs to specify the initial state and the temporal probabilities of the nodes. After specification, the validation mode can be selected to follow the changes of the system over time by browsing between time steps (only forward steps are supported) or by setting the number of steps and then plotting a graph of the variables. The DBN does not unroll graphically, only the values change. It is possible to set evidence by hand or by importing a data file. Of the inference methods discussed, only filtering and prediction are possible.

2.4.4.3 Netica

Netica is a product of Norsys (http://www.norsys.com), which is located in Canada. In Netica, the user designs the BN and then compiles it, after which inference is possible. Compiling the BN basically means that it is converted to a junction tree representation. When designing a DBN, temporal arcs can be added between nodes. The temporal arcs have a red color. The difference with BayesiaLab is that we explicitly define the temporal relationship (called a time-delay in Netica) of two nodes in one of the nodes instead of the implicit approach of BayesiaLab. In this way, *k*-order Markov processes can be modeled. When the DBN has nodes with temporal relations to itself (which is usually the case), the DBN definition contains loops. Of course, this is only

valid in the definition of the DBN; the unrolled version should not contain loops. After defining the DBN, it can be unrolled for *t* slices and compiled. The resulting BN is opened in a new window. Inference can be performed on this unrolled and compiled BN. All inference methods are supported. We can enter evidence by hand or by importing a data file.

2.5 Related work

Most of the work that is being discussed in this section focuses at DBNs usage to model the gene expression data. Incorporating prior knowledge parameters has not been touched in either of the discussed works.

In their research article *Modeling gene expression data using DBNs*, Kevin Murphy and Saira Mian have showed that most of the proposed discrete time models — like the Boolean network model, the linear model of D'haeseleer et al., and the nonlinear model of Weaver et al. - are all special cases of a general class of models called Dynamic Bayesian Networks (DBNs). Ability to model stochasticity, incorporating prior knowledge, handling hidden variables and missing data in a principled way are some of the advantages of DBNs (Murphy K & Mian S, 1999).

Yu Zhang et al presented a new dynamic Bayesian network (DBN) framework to model gene relationships by embedding with structural expectation maximization (SEM). Time-series data analysis and ability to deal with cyclical structures that cannot be achieved using their model. Regulatory network and the metabolic pathway were learned from *Saccharomyces Cerevisiae* cell cycle gene expression data, using this approach. Missing value processing in expression datasets and improvement in inference accuracy was derived as the output.

Many of the glitches while analyzing data during obtaining genetic regulatory networks from microarray data have been ignored. The rate of success depends on generalizing the current algorithms and gathering data in a simplified fashion with low measurement error and variance from single or multiple cells, makes the job easier for data analysts. No fundamental obstacles are seen to gather data of that kind which might enhance the performance of current Bayesian network discovery algorithms (Peter Spirtes et al., 2001).

Paul Helman et al proposed new technique to solve the problem of classifying gene expression data. Addressing classification enabled them to develop techniques that solved complexities of learning Bayesian nets. This model reduced the Bayesian network learning problem into a problem of learning multiple sub networks, each consisting of a class label node and its set of parent genes. Paul Helman et al's model is more appropriate for the gene expression domain than other structurally similar Bayesian network classification models like Naive Bayes and Tree Augmented Naive Bayes because of its consistency with prior domain experience assuming that small numbers of genes in different combinations are required to predict most clinical classes of interest. Two different approaches are considered to identify parent sets – one, that employs a greedy algorithm to search all genes; the second employs a gene selection algorithm whose results are incorporated as a prior to enable an exhaustive search for parent sets over a restricted universe of genes. Two other significant contributions are the construction of classifiers from multiple Bayesian hypotheses and algorithmic methods for binning gene expression data. Classifiers were validated using out-of-sample test sets

with a classification rate in excess of 90% on out-of-sample test sets for two publicly available datasets (Paul Helman et al., 2004).

Transcriptional networks from highly replicated gene expression profiling data have been reverse engineered using State-space models (SSMs). These are a class of DBNs in which the observed measurements are dependent on hidden state variables from Markovian dynamics, which can capture effects that are unable to be measured in a gene expression profiling experiment. (Matthew J Beal et al). Problem of inferring the model structure of these state-space models using both classical and Bayesian methods is addressed using this approach. Analogous model selection task in Bayesian theory is achieved by the usage of variational approximations. Certain interactions are present in both the classical and the Bayesian analyses of these regulatory networks, which are key for clonal expansion and for controlling the long-term behavior.

For interpreting large data sets produced during biological experiments, prior biological knowledge for the analysis would be very useful. Bayesian Decomposition is a tool that is capable of doing this task (TD Moloshok et al) with two advantages namely the ability to assign genes to multiple groups and encoding biological knowledge in to the system.

Microarray experiments that yield data about direction of interactions are usually two kinds: time series and perturbation experiments. To handle them correctly, the basic formalism should be modified (Norbert Dojer et al). The framework of DBNs to incorporate perturbations was applied while working with time series expression data. In addition, an algorithm for inferring network and a discretization method exclusive for time series data was introduced. This procedure has been applied to realistic simulations

data and the outcomes were compared to those obtained by standard DBN learning. Also, the advantages with using exact learning algorithm instead of heuristic methods were analyzed.

2.6 Hypothesis

The most challenging task in Bayesian simulations is the accumulation of proper data and their efficient declaration and incorporation into the model thereby assessing the prediction performance. This data is downloaded from the publicly available databases. The prior knowledge parameters are also downloaded in similar fashion from publicly available databases. After performing the data pre-processing, the experiments and the genes that satisfy the set-threshold were considered. Scores calculated using research literature was identified. These scores along with the expression data are combined and are simulated using BayesServer at different time-point combinations. The parameter learning and data distribution phases are adjusted in the GUI of the BayesServer software. At different number of parameter learning iterations, the model's prediction performances were observed and are plotted to calculate the correlation values of the simulation model.

3. METHODOLOGY

Understanding the effect of certain gene parameters when included in the time-course DBN model takes different paths as shown in Figure 3.1. The process starts with extracting the expression data, pre-processing, computing the correlation coefficients, choosing the right genes and experiments that have a correlation within the desired threshold, extracting the gene parameters (here the Gene Characterization Index (GCI) and the Gencards Inferred Functionality Score (GIFtS)) for the selected genes, loading the expression data with these parameters and finally, simulating the model that is built using the Bayes Server software. Each step is explained in detail in the following sections.

3.1. Data collection and preprocessing

From the gene expression omnibus (GEO) database, oligoarray data on the *"Prediction of toxicant-specific gene expression signatures following chemotherapeutic treatment of breast cell lines (GSE1647)"* were considered for this study (Troester, et al., 2004). Gene expression profiling has made it clear that the predominant cellular response to a variety of toxicants is stress response. This response commonly includes repression of protein synthesis, cell cycle regulated genes and induction of DNA damage. Troester et al lab has characterized the general stress response of breast cell lines derived from basal-like and luminal epithelium after treating them with doxorubicin (DOX) or 5-fluorouracil (5FU) and proved that each cell type has a distinct response. However, the researcher's assumption was that the expression changes induced by DOX and 5FU were unique to each compound and might explain the effect of these agents. So, significance analysis of microarrays to identify differentially expressed genes between DOX- and

5FU-treated cell lines was used. Followed by cross-validation analyses, the authors have found the genes that afforded high predictive accuracy. To test their accuracy, the cell lines were treated again with etoposide, a chemical similar to DOX. The outcome was that, using expression patterns of 100 genes, they were able to obtain 100% predictive accuracy in classifying the etoposide samples. This has proved that toxicant specific gene expression patterns vary according to cell type.

It is a known fact that DNA micro-arrays allow parallel expression profiling of thousands of genes. Manufacturing micro-arrays involves the synthesis of DNA onto a solid support or the deposition of pre-synthesized DNA onto a suitable surface. During the process of making the micro arrays, DNA is in the form of either PCR products or in the form of an oligonucleotide (Rouillard, Zuker, Gulari, 2003). DNA micro arrays that are made with DNA in the form of oligonucleotides are referred to as an oligoarray (Paul Helman et al., 2004).

The data we used for this simulation study consisted of breast cancer cell line information treated with toxic chemicals from 83 experiments across three time-course data points at 12-hour, 24-hour, and 36-hour intervals respectively. A total of 20,164 genes were present in the data array. Since the expression values of the majority of genes in the data between any two given experiments should be constant, data normalization techniques were adapted for pre-processing the data. When an experiment is performed on an oligonucleotide array, any sources of variation that are of non-biological origin should be eliminated without fail (Bolstad, et al., 2003). For this reason, normalization was applied on the high-noise oligoarray expression datasets. Reference sets are essential for any normalization process. Usually, the whole data set is considered as the reference

set, under the assumption that there is little variation in the genes being considered across the experiments, and the up and down regulated expression values are approximately symmetric (Calza, Valentini, Pawitan, 2008).

Normalization techniques are usually employed to minimize systematic variations and identifying biological differences (Karla & Grill, 2004). There are several normalizing methods available for the affymetrix data. Affymetrix data assume that the intensity in the array should be scaled to reach an average value. For our data, we have employed the Cyclic Loess Normalization technique that is available through the R packages from bioconductor (Bolstad, et al., 2003).

Loess is a method of non-linear logistic regression on the concept of M versus A plot, where M is the different values of log expression and A is the average of log expression values. Instead of considering the probe intensities of the same array, as in the traditional affymetrix data normalization steps, Loess oligo array normalization considers the gene expression values of two arrays at the same time by applying a correction factor from the curve obtained through the MA plot. Other normalization methods assume that

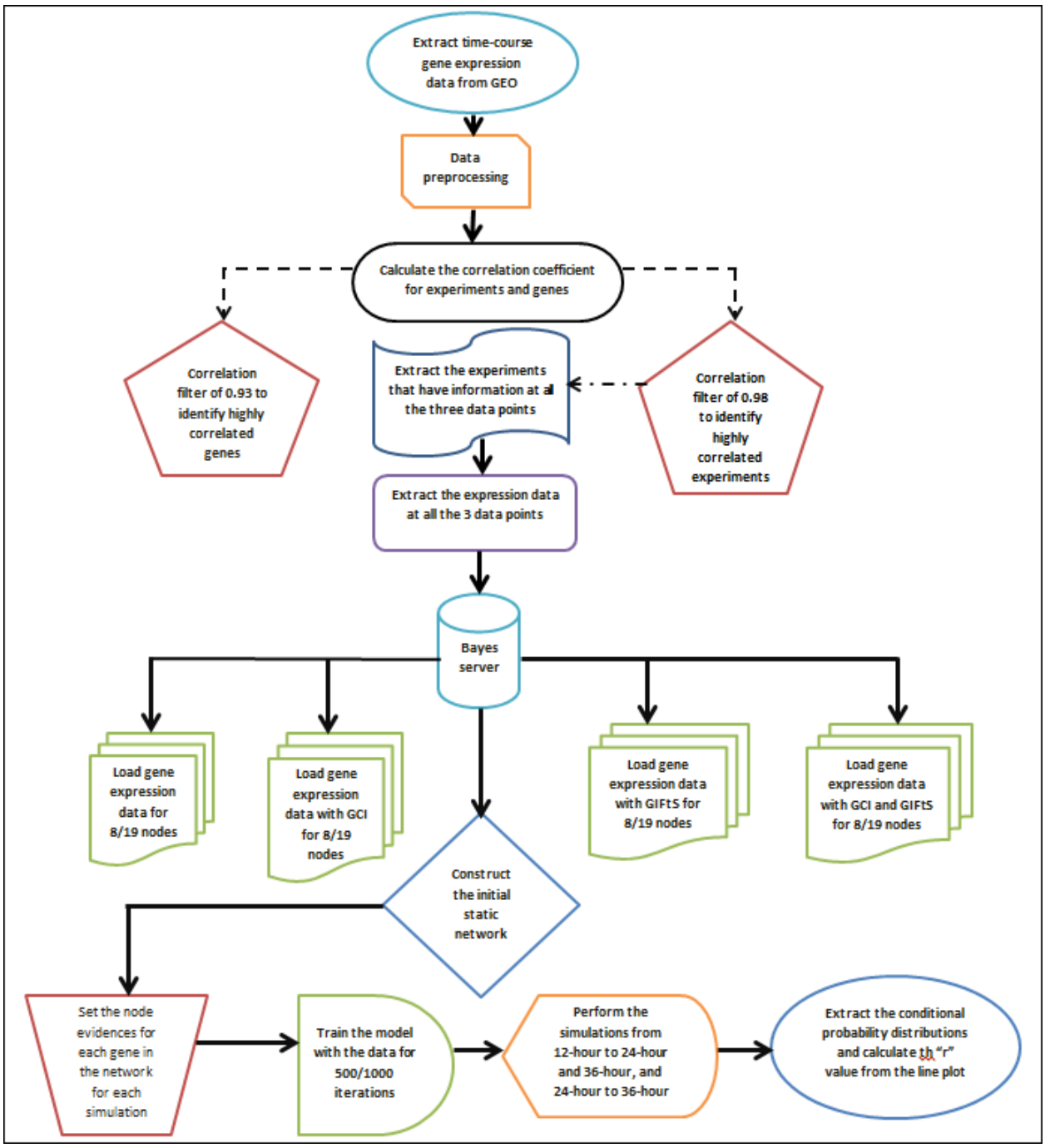

Figure 3.1: Flowchart of incorporating a priori knowledge into a DBN model for predicting time-course expression data.

expression levels of the genes do not change under experimental conditions (Karla & Grill, 2004), which is not true for our dataset. Cyclic Loess method preserves row means of the data matrix and adds a constant if there is an increase or decrease in the column values. On these reasons, we have adapted the Loess normalization technique to reduce noise in our data. For two arrays *i* and *j* with gene expression values X_i and X_j, the logistic regression is calculated as $M_k = \log_2 (X_i / X_j)$ and $A_k = ½ \log_2 (X_i X_j)$ [8]. A Loess normalization curve is fitted using the M versus A plot. The process is repeated for all the pairs in the experimental dataset until a good curve fitting is achieved. Usually, one or two iterations per pair will yield the best results.

The following MA plot gives an overview of the data before and after normalization. The plot explains the distribution of the data and its' skewness.

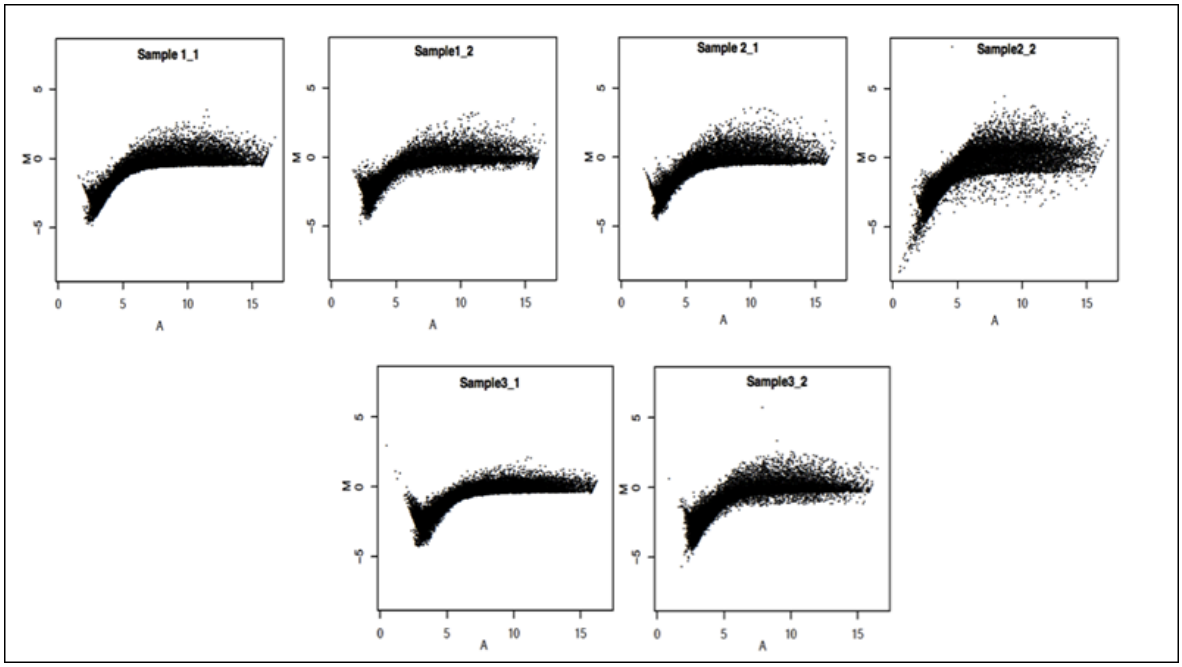

Figure 3.2 (a): Data distribution before normalization. Sample 1- At 12h, Sample 2 – At 24h, Sample 3 – At 36h

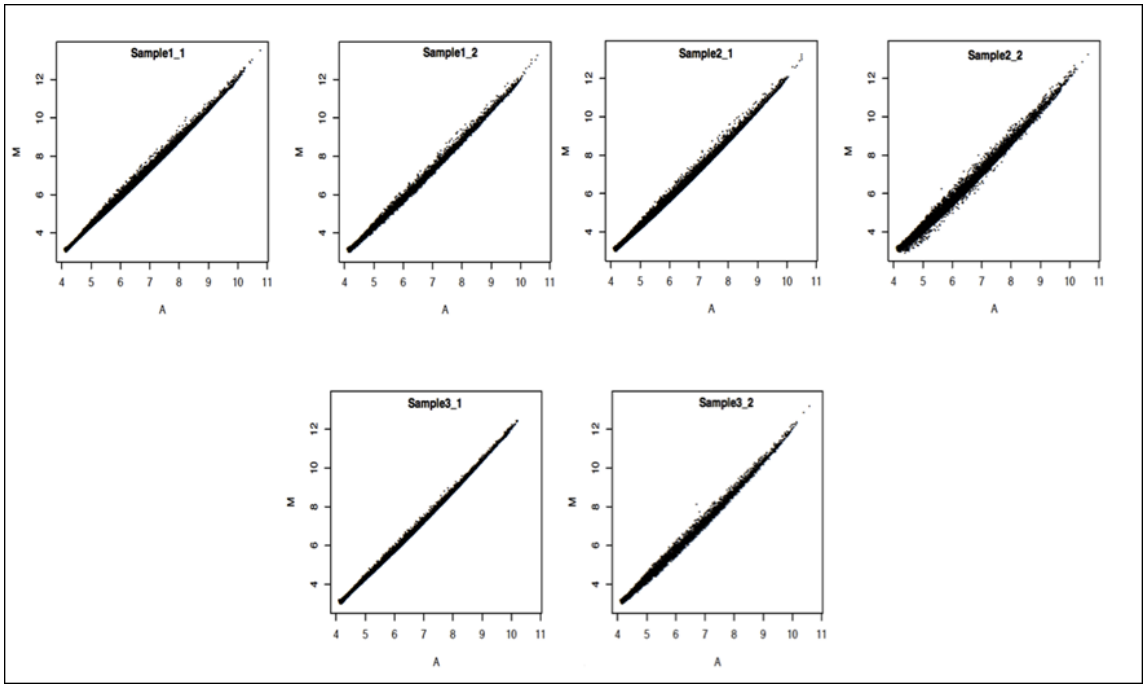

Figure 3.2 (b): Data distribution after normalization. Sample 1- At 12h, Sample 2 – At 24h, Sample 3 – At 36h

3.2. Correlation computation

In any Bayesian model, it is essential to know how closely the variables in the model are related. It is the mutual communication and the inter-dependency of the variables that bring out the best predictions through the simulations that are carried out in a dynamic Bayesian model.

To identify the closely related genes from the list of 20,164 genes in the dataset, we calculated the Pearson's correlation coefficient, r (Rogers & Girolami, 2005), where X is the first gene or experiment data and Y is the other gene or experiment data, between the experiments and also for all the genes. Experimental correlation provided us the information about which experiments had a linear relationship. Correlations among the

genes provide the information about the genes that are closely related and follow a linear relationship. The correlation was calculated using the following formula in Figure 3.3, and the correlation values fell into the range from 0.958 to -0.907.

$$r = \frac{\sum XY - \frac{\sum X \sum Y}{N}}{\sqrt{(\sum X^2 - \frac{(\sum X)^2}{N})(\sum Y^2 - \frac{(\sum Y)^2}{N})}}$$

Figure 3.3: Formula to calculate Pearson's correlation coefficient.

A correlation cut-off value of 0.93 was chosen to identify the genes that are highly inter-dependent. Eight genes were found to be highly related after eliminating the duplicates. For the initial test run with 500 iterations, these eight genes were considered. At a threshold of 0.9, 19 genes were found to be highly correlated that were used for the 1000 iteration simulation model. Experiments that exhibited a correlation of greater than 0.98 were considered from the 83 experiments in the dataset. Experiments that are common across all the three time-course data points and that possessed a correlation value of greater than or equal to 0.98 were chosen for the simulation.

3.3 Gene Characterization Index (GCI)

Gene annotation is vital in computational genomics that deal with expression analysis, prediction of gene function and in-depth observation into the sequence (Kim & Miyano, 2004). The gene characterization index (GCI) accounts to the characterization status of individual genes, which in turn, explains how far the protein-encoding gene is

functionally described (Kemmer, 2008). Using GCI as a parameter in the Bayesian model adds weight in the form of gene annotation to the genes that are present in the model.

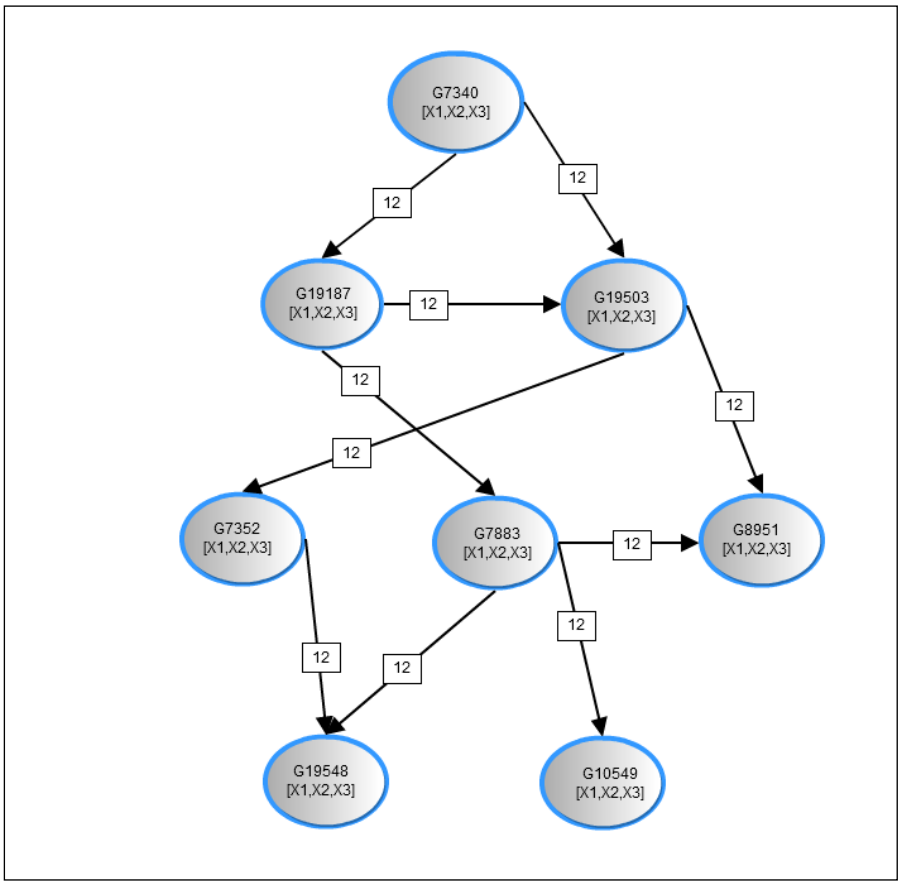

Figure 3.4: Static Bayesian Network constructed using BayesServer at 12-hour time point. X1, X2 and X3 represent the expression data, GCI and GIFTS score respectively.

The scoring procedure of assigning a GCI score to each gene in the human genome is based on the responses to a global survey by 50 researchers worldwide on a scale of 1 to 10 with 1 being poor and 10 being extensive (GCI). Genes for the survey were chosen from major online resources like Entrez Gene and GeneLynx databases (Lenhard, Hayes and Wasserman, 2001). Figure 3.5 indicates the GCI scores of the genes in the human genome across three different time points. This made it possible for us to

incorporate GCI scores into the DBN model. The GCI score will help us to identify groups of genes with potentially interesting applications, which are usually overlooked by the scientific community. By incorporating the GCI score in our Bayesian model, in addition to the annotation value of the gene, the confidence score of the gene for being involved in breast cancer is assigned. Another major reason for incorporating the GCI score in our Bayesian model is to serve the purpose of being a resource for scientists to demonstrate the novelty of research findings and utility of new methods.

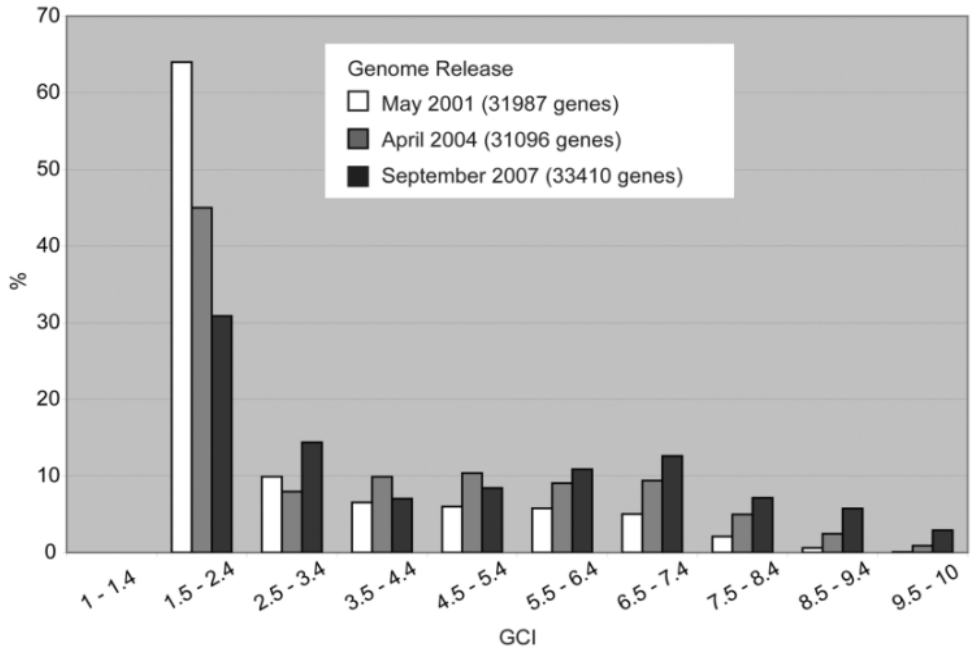

Figure 3.5: Histogram displaying the frequency of GCI scores observed in the analysis of genes at 3 different time points after the release of the first draft of the human genome sequence. (Kemmer et al, 2008)

3.4. GenCards Inferred Functionality Score (GIFts)

GenCards is a resource of information on annotations of over 50,000 human gene entries that is built upon 68 data sources including Gene Ontology, pathways, interactions and phenotypes. GIFtS is usually obtained by submitting the gene symbols to the GenCards database. Genbank accession numbers in the dataset are converted into their respective gene symbols using bioDBnet (Mudunuri, 2009).

To identify the DNA sequence's function, scientists have used different approaches involving the assessment of annotation spectrum of a given gene. Some efforts in this regard include the Genome Annotation Scores (GAS) algorithm, GO Annotation Quality (GAQ) and also the Gene Characterization Index (GCI) which scores the extent to which a gene is functionally described (Rouillard, Zuker, Gulari, 2003).

GIFtS utilizes the wealth of gene annotations within GenCards which is a rich source of over 50,000 human gene entries that gets information from about 70 different sources. As quoted, *"GIFtS value for a gene is defined as the number of GenCards sources, out of 68."* (Harel et al., 2009). Genes falling in the high GIFtS region account for higher probability and those in the lower regions indicate less chances of being associated with relevant biological changes. A binary vector of 67 elements for each gene, indicating presence or absence of data in each relevant source was identified from literature.

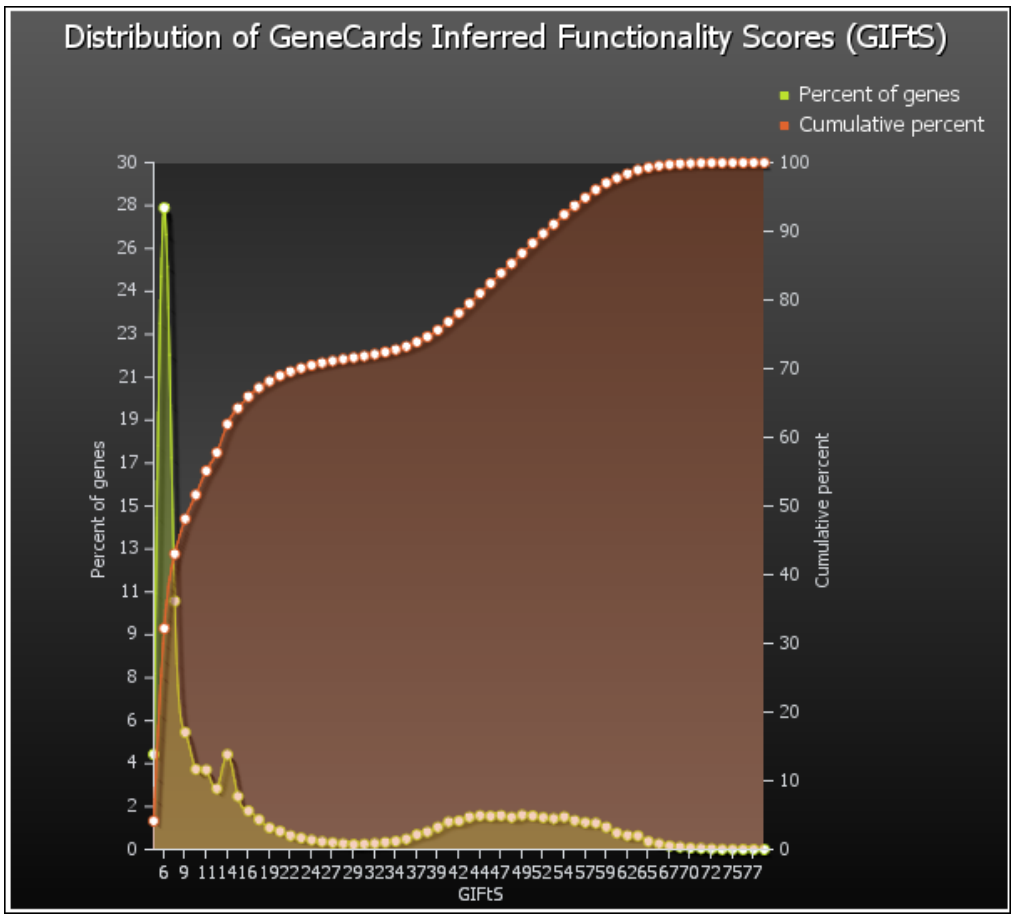

Figure 3.6: GIFtS score distribution with the percentage of genes on Y-axis and the score on x-axis. (Harel et al, 2009)

Gene Numbers	Swissprot ID	GCI	GIFtS
G7340	TBB5_HUMAN	10	67
G7352	TBB1_HUMAN	8.9	61
G7883	CDN1A_HUMAN	9.8	69
G8951	PLAK_HUMAN	9.8	66
G10549	ID3_HUMAN	8.8	61
G19187	CDN2A_HUMAN	9.9	69
G19503	PTTG1_HUMAN	9	61
G19548	KIF3A_HUMAN	8.9	60

Table 3.1: Genes after applying a correlation threshold of 0.93.

3.5 Simulations

After applying the correlation filter with a threshold of 0.93, a set of 8 genes as listed in Table 3.1 was obtained. As an initial test run, the pathway of interactions among these 8 genes was identified using KEGG pathway database (Kanehisa, 2002) and Metacore (Yang, et al., 1998) and a static Bayesian network was constructed using the 12-hour expression data.

In the Bayes server software, the GCI and the GIFtS scores were set as the node parameters and the 24-hour expression data was set as the node evidence for the simulation to occur. This static Bayesian network was trained for 500 iterations and the simulation was initiated. Once the simulation was completed, the conditional probability distributions of each node in the 24-hour dynamic Bayesian network were recorded and compared to the 12-hour static Bayesian network. This process is repeated for the 12-hour to 36-hour simulations and for 24-hour to 36-hour simulations. For each simulation, the time course into which the current static network is being simulated is set as the node evidence. A total of 12 different data combinations are considered for each simulation. Prediction capability of all the simulations is evaluated using regression method and obtaining the r-value. The model showed a high r-value for the 12-hour to 24-hour simulation with only gene expression being considered for the prediction, followed by the combination of gene expression, GIFtS and gene expression with the GCI score.

Two experiments were carried out in this study. One with the 8-gene set at 500 by using a correlation threshold filter of 0.93 and the second one with the 19-gene set at 300, 500, 700 and 1000 iterations obtained using a threshold value of 0.90. When the

correlation threshold is changed to 0.90 from 0.93, 19 nodes were obtained that satisfied the condition, which also included the earlier 8 nodes. As a whole, a total of 11 nodes were added to the initial 8-gene set.

Table 3.2 summarizes the data combinations, parameters, gene sets and time points that are involved with the simulation.

Parameter	No. of Parameters	Information
Type of gene sets	2	8-gene set and 19-geneset
Node parameters	3	Gene expression, GCI and GIFts
No. of Simulations	3	12-hour to 24-hour, 12-hour to 36-hour and 24-hour to 36-hours
Time points	3	12-hour, 24-hour and 36-hours
Iterations for learning	4	300, 500, 700 and 1000
Data Combinations	4	Only Gene Expression (GE), GE-GCI, GE-GIFts and GE-GCI-GIFts

Table 3.2: Summary of simulation information

4. RESULTS

The results of the simulation of breast cancer expression data using dynamic Bayesian models to assess their prediction performance are being discussed in this section. The outcomes obtained in different experiments are explained here. Experiment 1 is carried out with 8-genes that were obtained using a correlation of 0.93 at 500 parameter learning iterations. Experiment 2 is carried out with 19-genes that were obtained using a correlation threshold of 0.90 at 300, 500, 700 and 1000 parameter learning iterations. Each of these iterations include four different data combinations called Simulation with only gene expression data, expression data with GCI score, expression data with GIFTS score and all the three score combined together.

The below figure indicates the static Bayesian network constructed from the information retrieved from KEGG and Metacore software. The static network remains the same devoid of the time points.

4.1 Experiment 1

This experiment was performed on a smaller sub set of gene set obtained from a gene correlation threshold of 0.93. The initial parameter learning was set at 500 iterations and the 12-hour static bayesian model was simulated into 24-hour and 36-hours along with the 24-hour static being simulated into 36-hour.

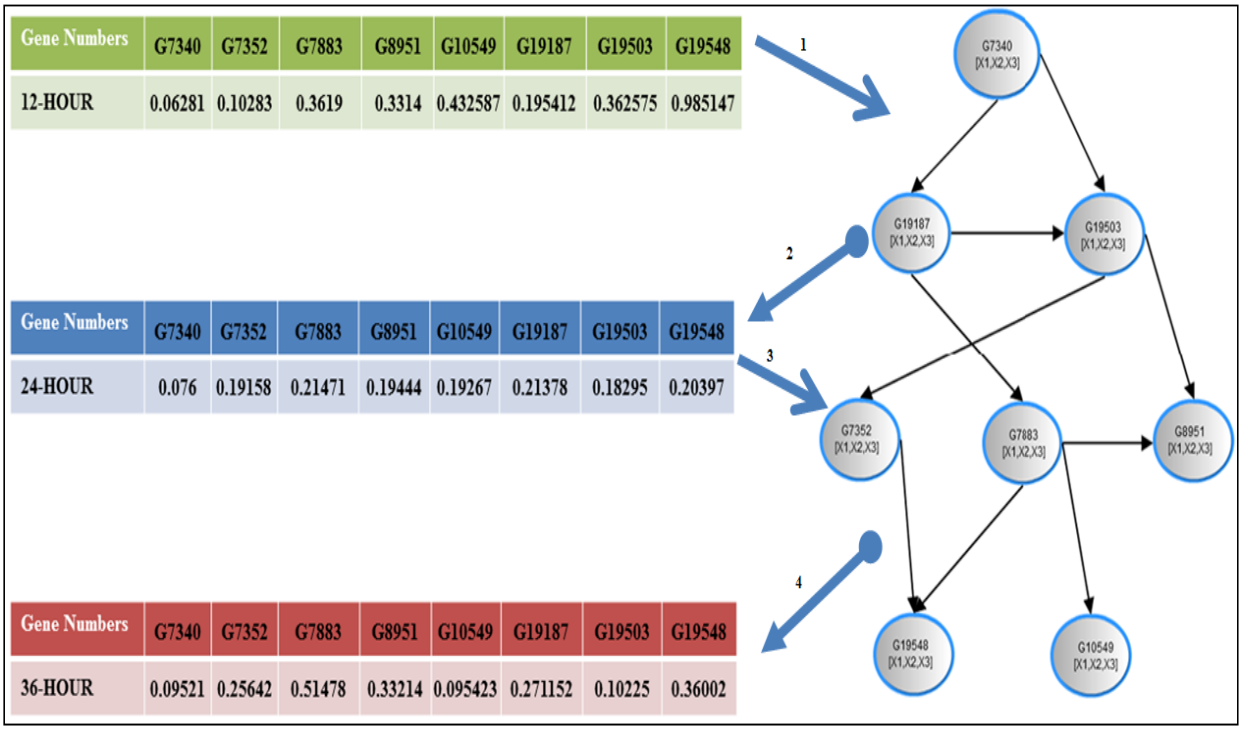

Figure 4.1: An overview of simulation process starting at one time-point and predicting the other. The table on the left indicate the scores of all the three parameters, Gene Expression, Gene Characterization Index (GCI) and Genecards Inferred Functionality Score (GIFtS), combined. The Bayesian network to the right is same at all time-points and is constructed using the Bayes server software. Normal arrows indicate the input data to the network. Circle-ended arrows indicate the time-point into which the simulation is predicting. Arrow 1 indicates the use of 12-hour data for 12-24 hour simulation. Circle-ended Arrow 2 indicates the prediction of 24-hour from 12-hour data. Arrow 3 indicates the use of 24-hour data for 24-36hour simulation. Circle-ended arrow 4 indicates the prediction of 36-hour from 24-hour data.

4.1.1 Gene expression data

4.1.1.1 12-hour to 24-hour simulation

As mentioned earlier before, the static 12-hour network was constructed using the information obtained from KEGG and Metacore [66]. The figure 4.2 below represents the 12-hour data being loaded into the nodes in BayesServer.

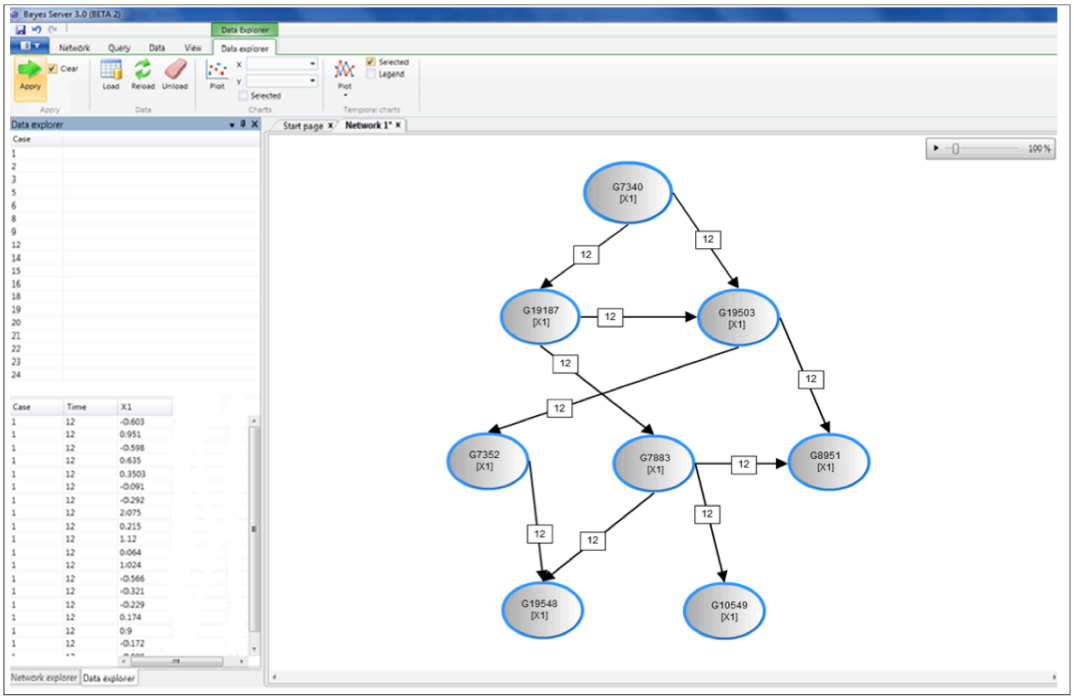

Figure 4.2: 12-hour static Bayesian network in BayesServer

This particular network is simulated into the 24-hour and there had been a couple of topological changes that were observed. In figure 4.3, the red color edge indicates the reversal of connection between G7340 and G19187 and a new connection was established between G7352 and G19187, G19548 and G10549, and G10549 and G8951. These changes are understood in a sense that the expression data when combined with other priori parameters leads to certain changes in the path of the information passage

leading to changes in the edges. The green edges in the below figure 4.3 indicates the changes in the edges appearance.

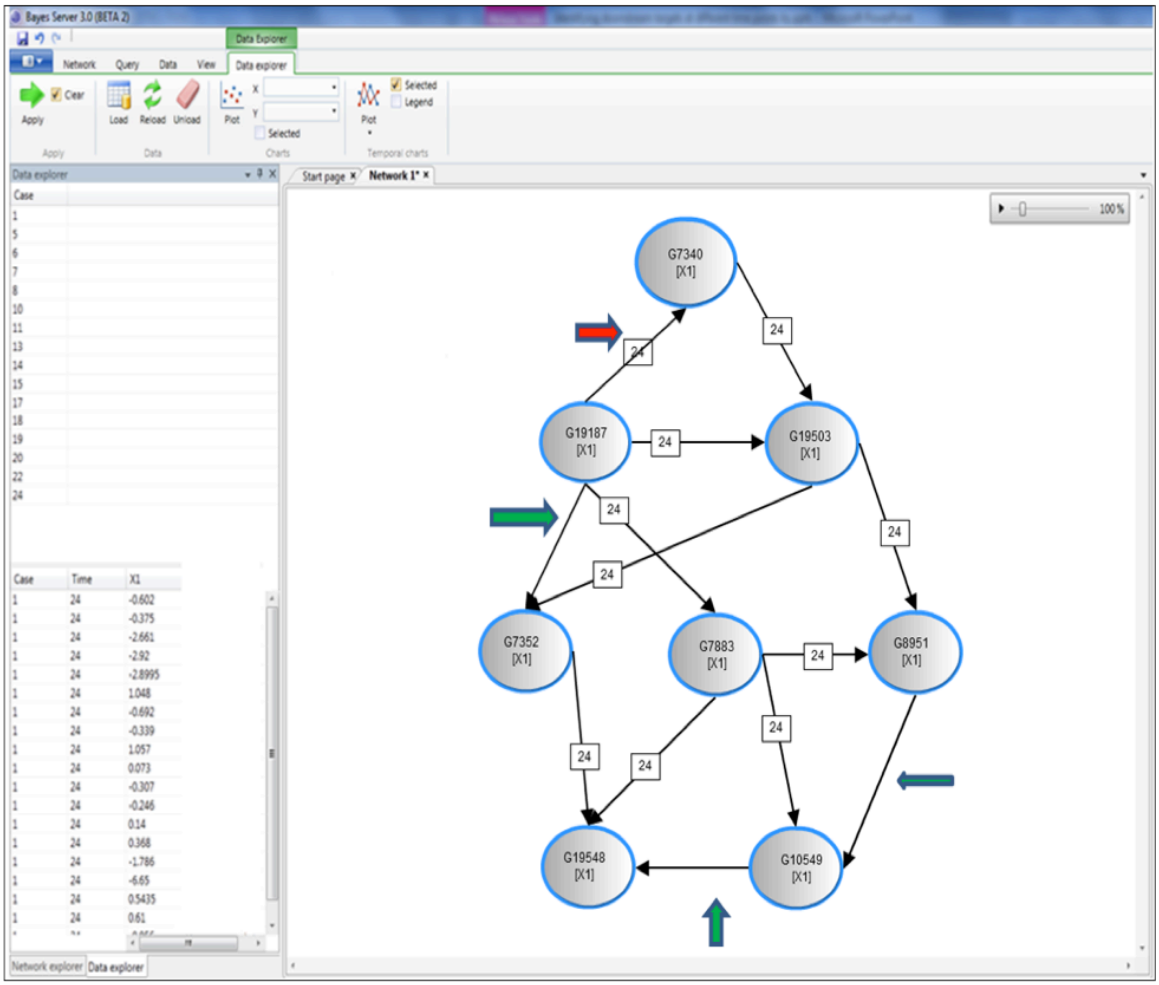

Figure 4.3: 12-hour static BN simulated to 24-hour time point

The node conditional probability distributions at each node at each time point and the model's prediction are presented in the table 4.1. A line plot of this particular simulation is also presented in figure 4.4.

Gene ID	Swissprot ID	CPD at Static 12h	CPD at DBN 24h	CPD at Static 24h
7340	TBB5_HUMAN	0.07795	**0.0853**	0.076
7352	TBB1_HUMAN	0.19096	**0.1896**	0.19158
10549	ID3_HUMAN	0.211	**0.367**	0.21471
7883	CDN1A_HUMAN	0.19632	**0.2874**	0.19444
19503	PTTG1_HUMAN	0.19356	**0.3852**	0.19267
19548	KIF3A_HUMAN	0.211	**0.1952**	0.21378
19187	CDN2A_HUMAN	0.1827	**0.1639**	0.18295
8951	PLAK_HUMAN	0.20902	**0.2861**	0.20397

Table 4.1: Node CPDs of the static 12-hour, 24-hour and the 24-hour network after simulations.

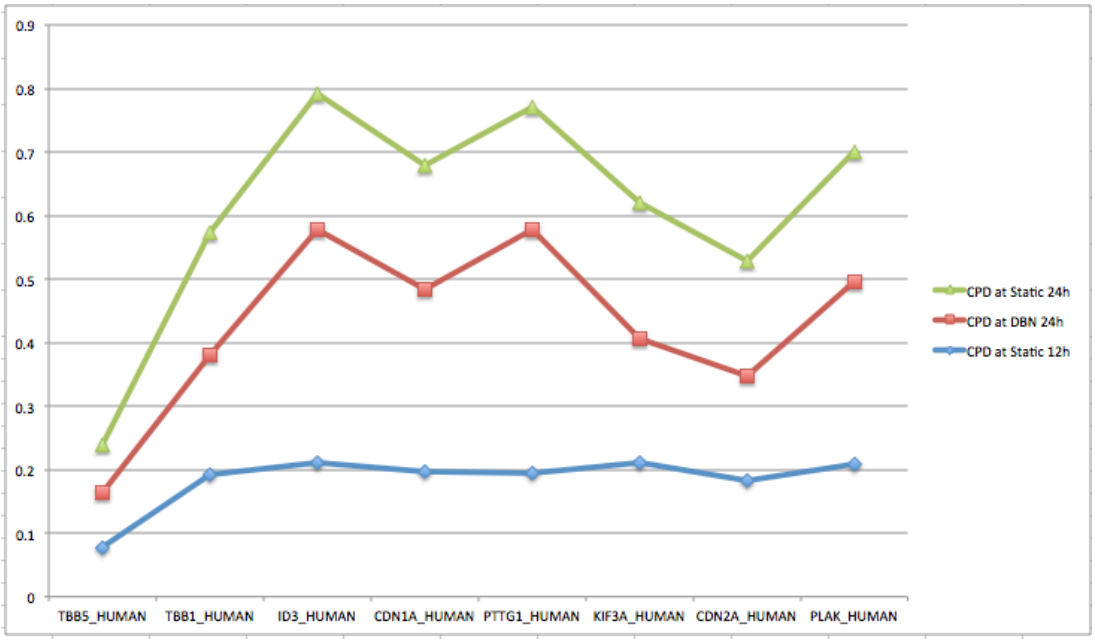

Figure 4.4: Line plot of expression data's 12-hour to 24-hour simulation. The plot indicates the model's accuracy to be 99%.

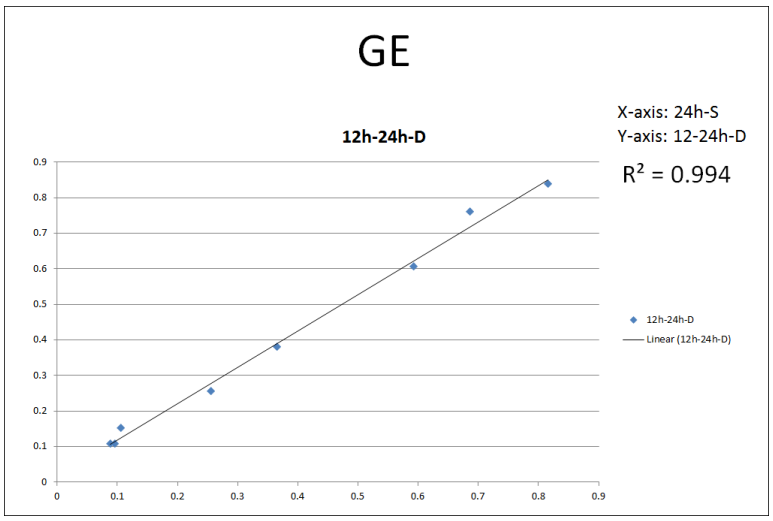

Figure 4.5: Dot plot of the node CPDs of 24-hour initial static network and the simulated 24-hour network

4.1.1.2 12-hour to 36-hour simulation

There are two edge reversals in the 12-hour to 36-hour simulation, both from G19187 and G19503 to G7340 and there are two edge deletions from G19503 and G7883 to G8951. An additional edge has come into the picture between G10549 and G8951.

Figure 4.6: Line plot of the expression data's 12-hour to 36-hour simulation. The plot indicates the model's accuracy to be 97%.

Gene ID	Swissprot ID	CPD at Static 12h	CPD at DBN 36h	CPD at Static 36h
7340	TBB5_HUMAN	0.07795	**0.08921**	0.078009
7352	TBB1_HUMAN	0.19096	**0.1743**	0.190638
10549	ID3_HUMAN	0.211	**0.2058**	0.214417
7883	CDN1A_HUMAN	0.19632	**0.21345**	0.193888
19503	PTTG1_HUMAN	0.19356	**0.18736**	0.194145
19548	KIF3A_HUMAN	0.211	**0.2314**	0.210715
19187	CDN2A_HUMAN	0.1827	**0.10583**	0.182903
8951	PLAK_HUMAN	0.20902	**0.11329**	0.208185

Table 4.2: Node CPDs of the static 12-hour, 36-hour and the 36-hour network after simulations.

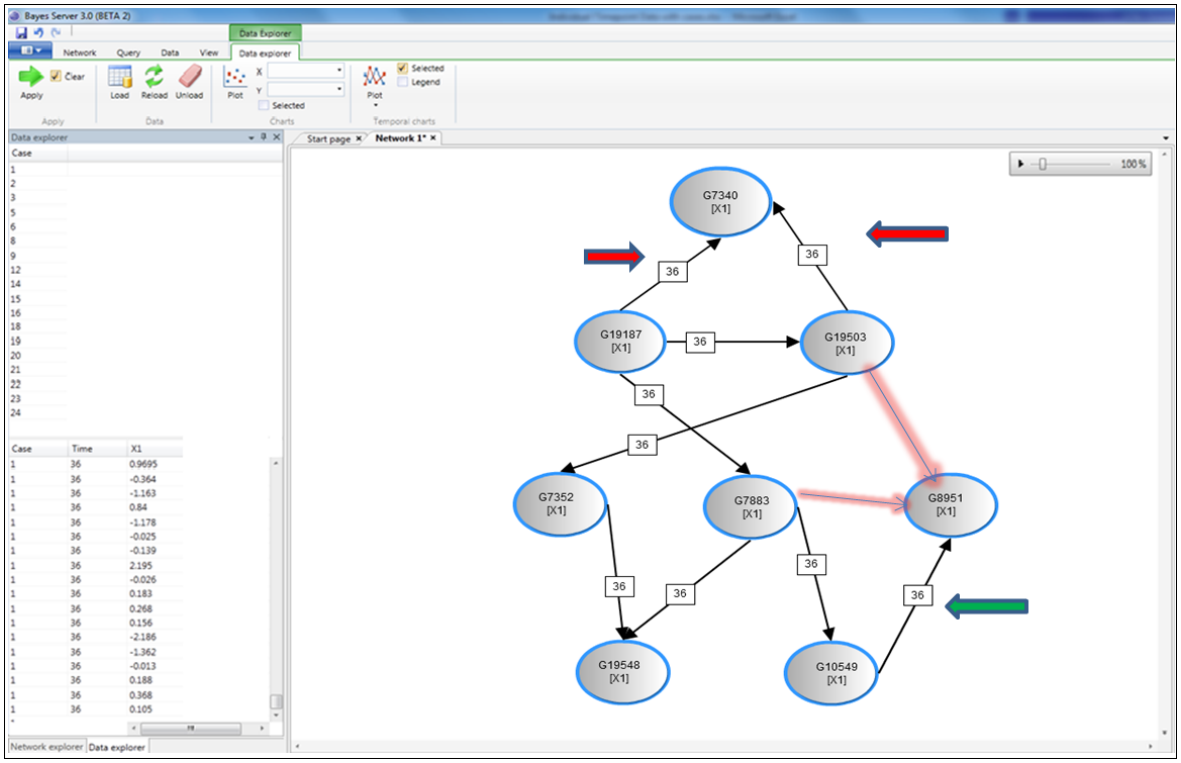

Figure 4.7: 12-hour static BN simulated to 36-hour time point

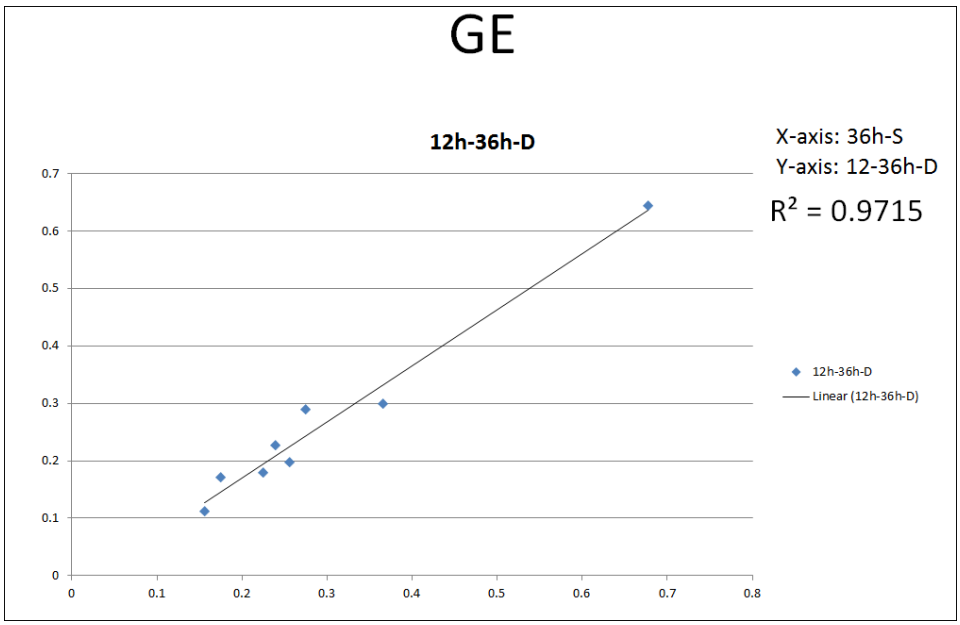

Figure 4.8: Dot plot of 12-hour static Bayesian network to 36-hour

4.1.1.3 24-hour to 36-hour simulation

The 24-hour to 36-hour simulation has showed an accuracy of 89%. The line plot is given in the figure 4.9.

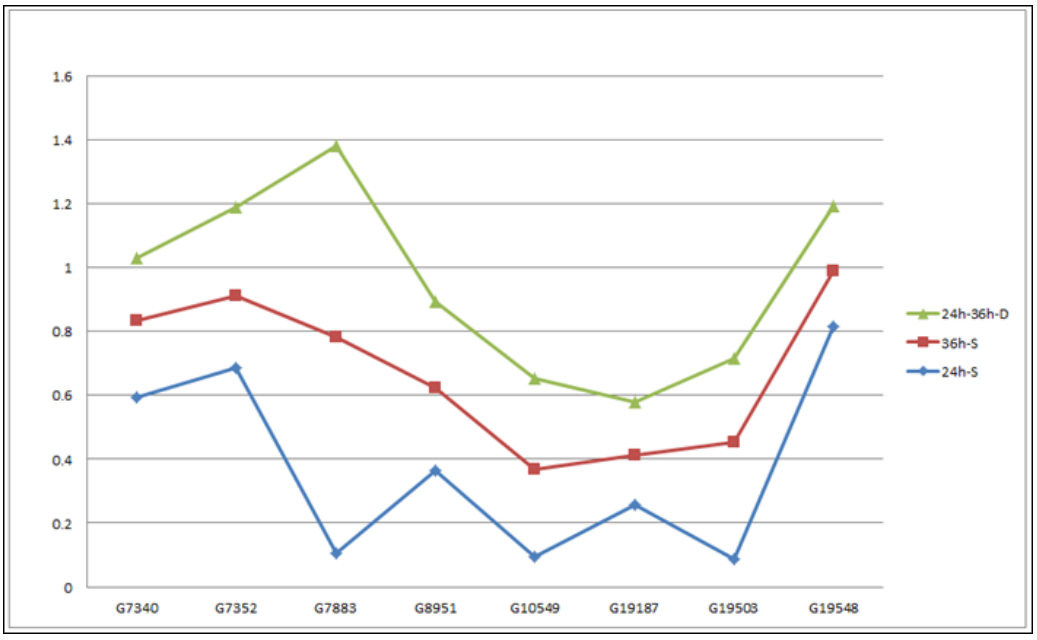

Figure 4.9: Line plot of the expression data's 12-hour to 36-hour simulation. The plot indicates the model's accuracy to be 91%.

Gene ID	Swissprot ID	CPD at Static 24h	CPD at DBN 36h	CPD at Static 36h
7340	TBB5_HUMAN	0.076	**0.07916**	0.078009
7352	TBB1_HUMAN	0.19158	**0.18535**	0.190638
10549	ID3_HUMAN	0.21471	**0.1995**	0.214417
7883	CDN1A_HUMAN	0.19444	**0.22675**	0.193888
19503	PTTG1_HUMAN	0.19267	**0.17421**	0.194145
19548	KIF3A_HUMAN	0.21378	**0.20727**	0.210715
19187	CDN2A_HUMAN	0.18295	**0.13583**	0.182903
8951	PLAK_HUMAN	0.20397	**0.20929**	0.208185

Table 4.3: Node CPDs of the static 24-hour, 36-hour and the 36-hour network after simulations.

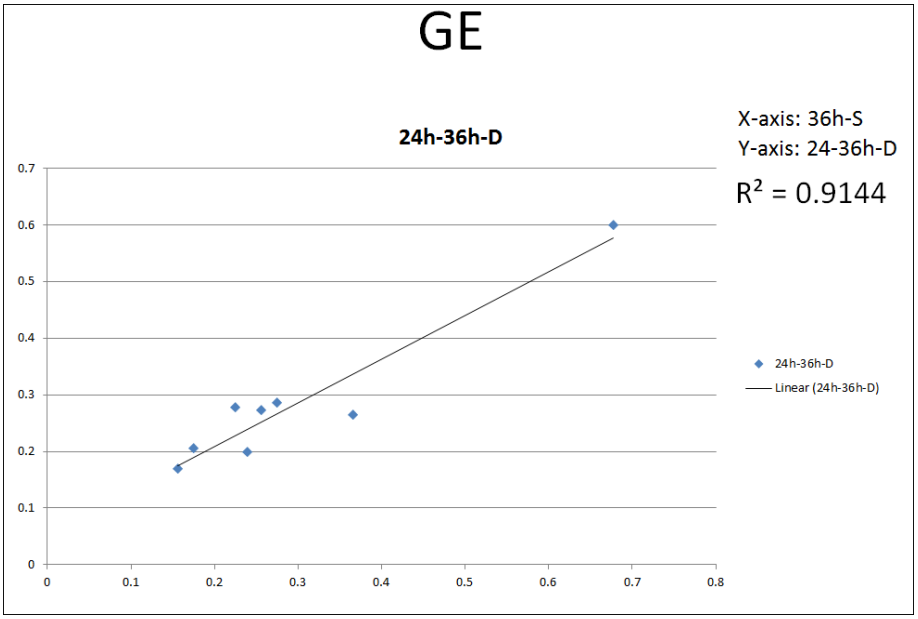

Figure 4.10: Dot plot of Dot plot of 24-hour static Bayesian network to 36-hour

4.1.2 Gene expression data with Gene Characterization Index (GCI)

Gene Characterization Index (GCI) is used in combination with the Gene expression data for all the three time point simulations. The initial 12-hour to 24-hour simulation has been found to have the best prediction accuracy of around 99%. More details about the simulation are presented in the further sections.

4.1.2.1 12-hour to 24-hour simulation

The 8-geneset's 500 iteration simulation of the gene expression and GCI score data combination recorded a 99% prediction performance. The dot plot of the prediction CPDs is presented in the figure 4.11.

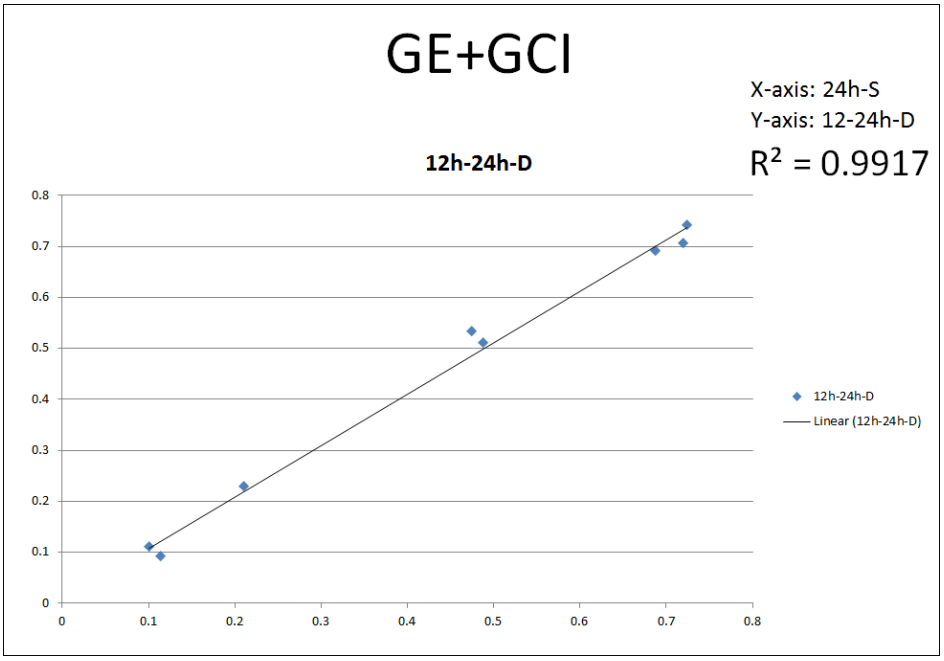

Figure 4.11: Dot plot of 12-hour static Bayesian network to 24-hour with Gene expression and GCI combined data set.

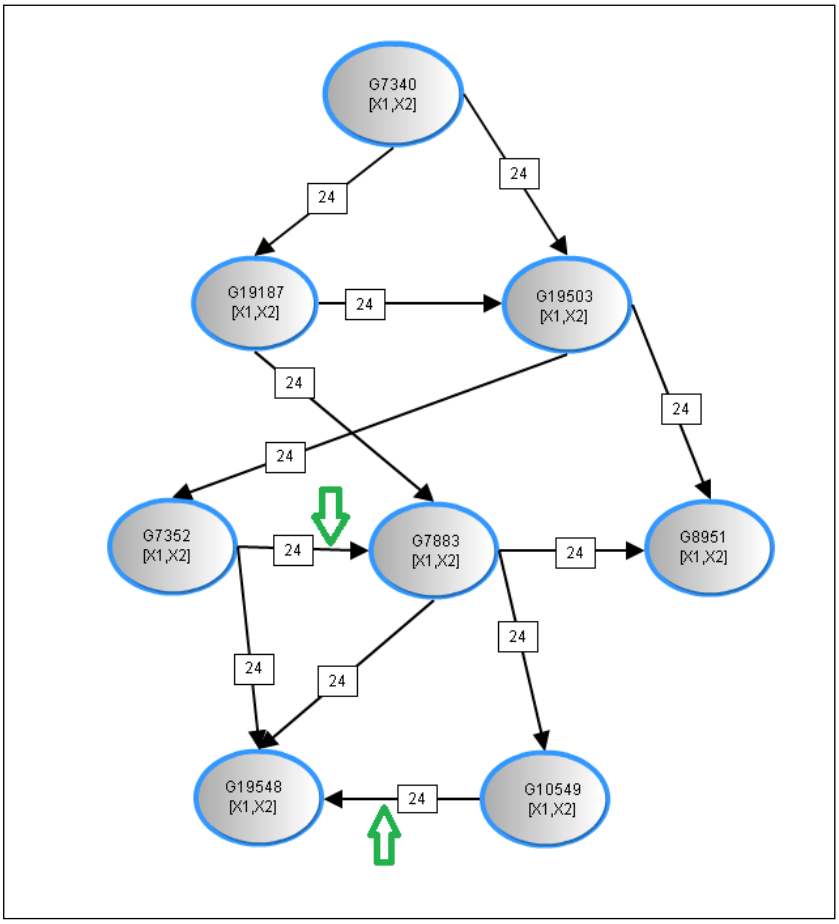

Figure 4.12: Edge changes during simulation of 12-hour to 24-hour using Gene expression and GCI score together

The Bayesian network obtained after the simulation is depicted in the figure 4.12. No edge deletions were observed except the addition of newer ones between G19548 and G10549 and G7352 and G7883 (indicated in green arrows).

4.1.2.2 12-hour to 36-hour simulation

This set of data and time point combination recorded a prediction performance of 95%. The reason for the downfall of prediction performance is the direct passage of

information between the genes at those time points. As 24-hour time point immediately follows the 12-hour, it surpassed the remaining simulations in most of the cases and 12-hour to 36-hour is recorded at low levels due to a shift in the time points from 12-hour to 36-hour instead of 24-hour. The dot plot in figure 4.13 indicates the distribution of CPDs.

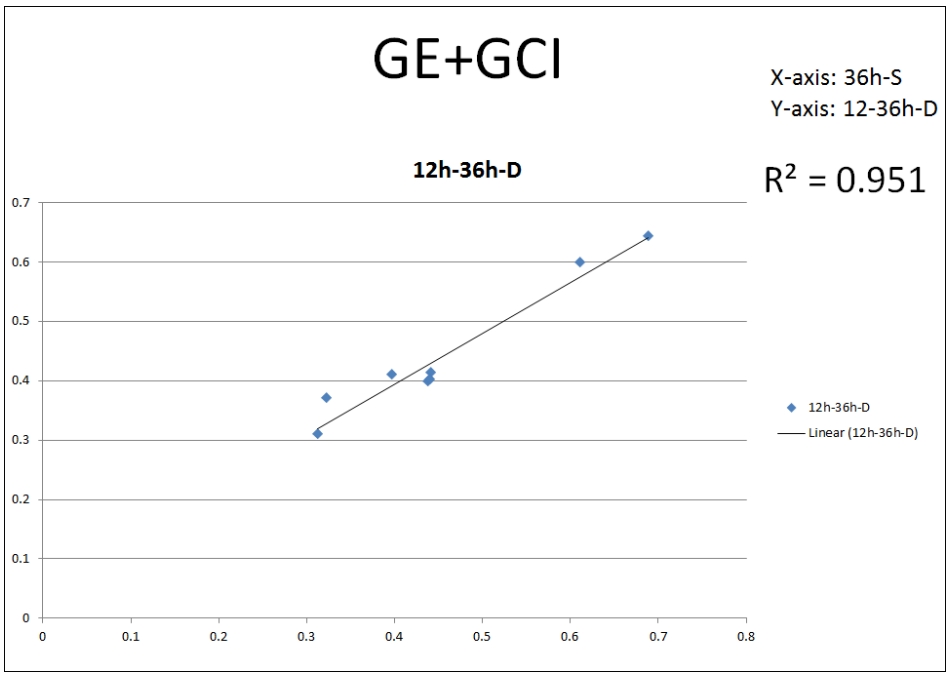

Figure 4.13: Dot plot of 12-hour static Bayesian network to 36-hour using gene expression and GCI score.

The final Bayesian network obtained after the simulation is represented in figure 4.14 indicating the edge addition between G7352 and G7883 and a deletion of edge between G19503 and G8951.

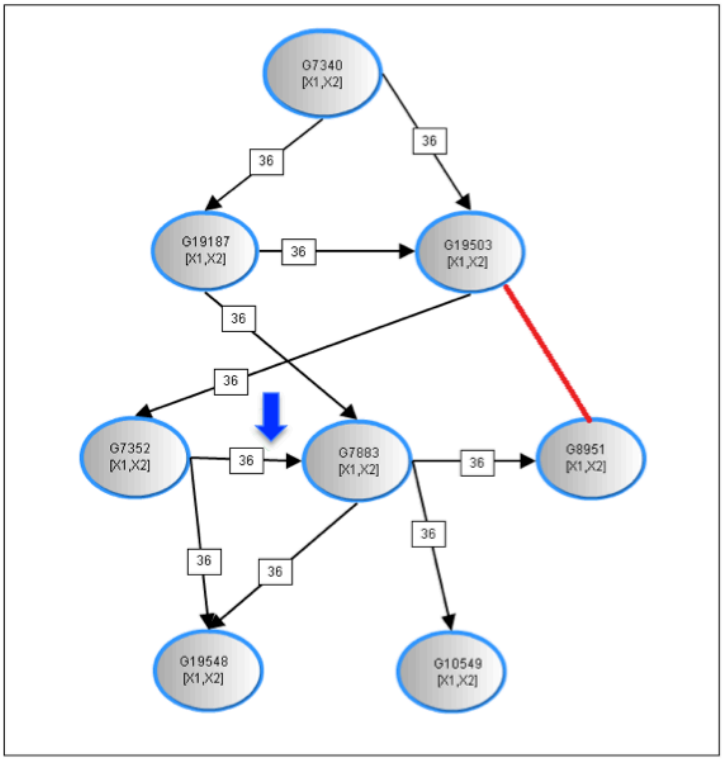

Figure 4.14: Edge changes during simulation. Red line indicates the deletion of initially present edge. Blue arrow indicates reversal of edge direction.

4.1.2.3 24-hour to 36-hour simulation

This simulation showed a prediction performance of 97% standing between the previous two simulations. To summarize the Gene expression and GCI data combination the initial simulation from 12-hour to 24-hour stood at the top place compared to the other two simulations.

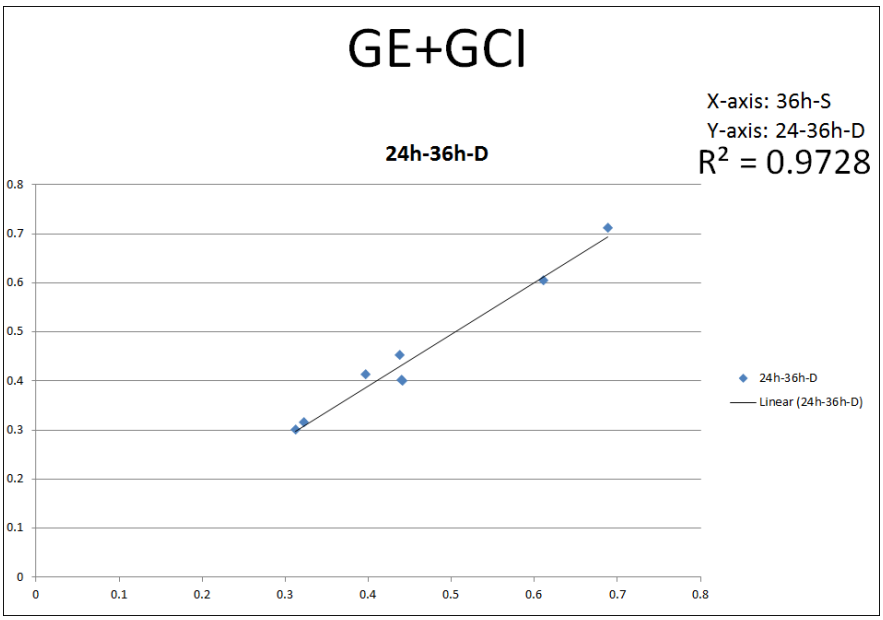

Figure 4.15: Dot plot of 24-hour to 36-hour simulations using expression data and GCI score.

4.1.3 Gene expression and Gencards Inferred Functionality Score (GIfTS)

This particular data combination is simulated for the same time point combinations with an average prediction performance of 97%, when all the three simulations are considered.

4.1.3.1 12-hour to 24-hour simulation

This simulation combination yielded an overall prediction performance of 99% followed by the 12-hour to 36-hour simulation and the last one. The line plots indicate the same thing in the below figure 4.16.

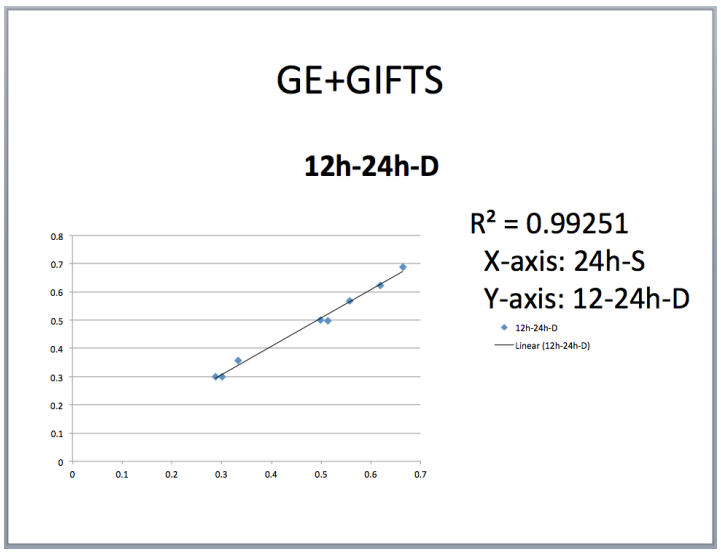

Figure 4.16: Dot plot of 12-hour to 24-hour simulation of expression data and GIFtS score combination.

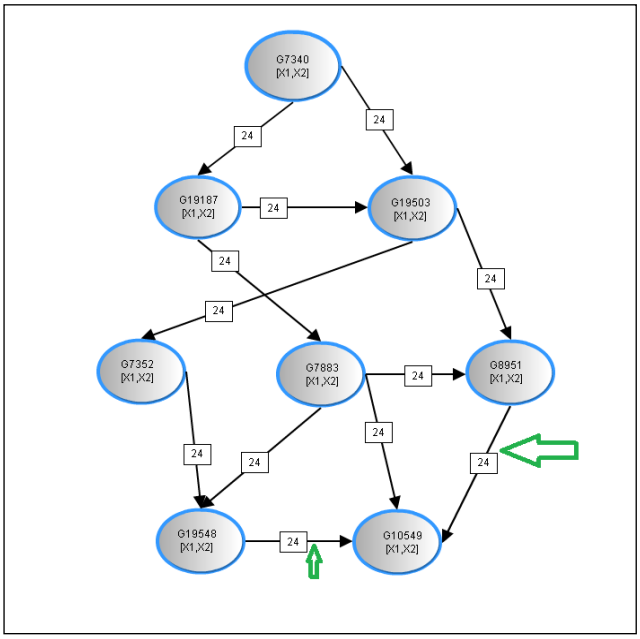

Figure 4.17: Edge changes in 12-hour to 24-hour GE and GIFtS combination simulation.

The Bayesian network obtained after the simulations has been presented above in figure 4.17, indicating the edge changes. There has been an additional edge between G19548 and G10549, G10549 and G8951.

4.1.3.2 12-hour to 36-hour simulation

This simulation recorded a prediction performance of 98%. The line plot indicates the distribution of node CPDs obtained through the simulation.

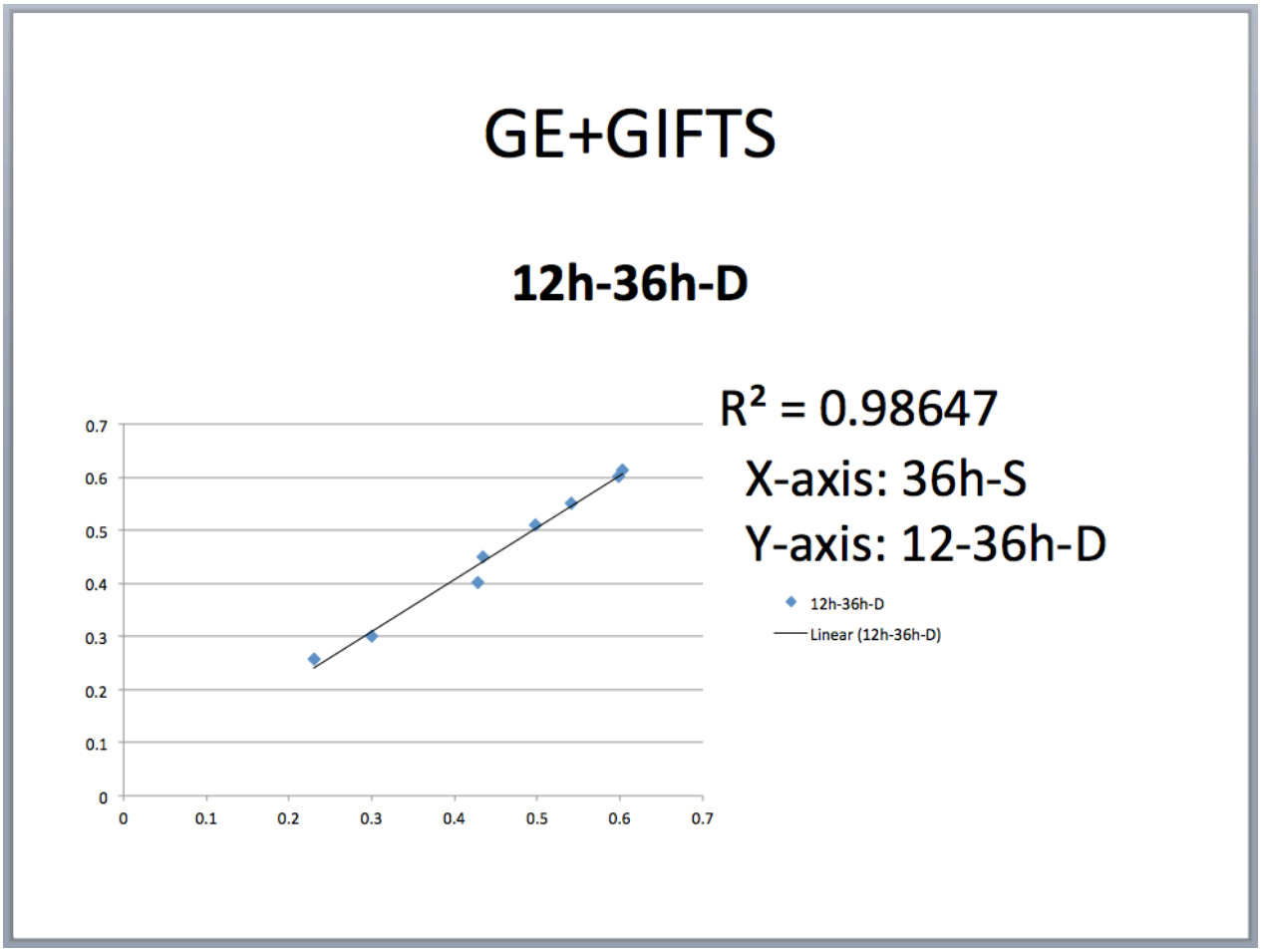

Figure 4.18: Dot plot of 12-hour to 36-hour Bayesian simulation of expression data and GIFtS score combined together.

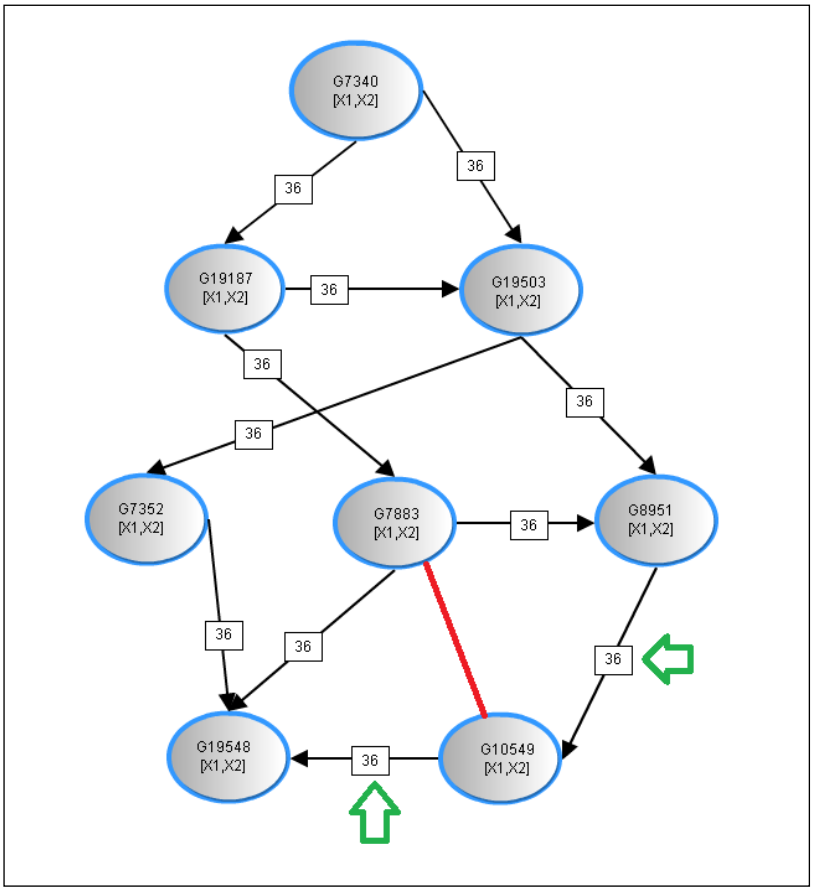

Figure 4.19: Edge changes in GE and GIFtS score combined simulation from 12-hour to 36-hour.

A new edge has been created between G19548 and G10549 and between G10549 and G8951 with an edge deletion between G7883 and G10549.

4.1.3.3 24-hour to 36-hour simulation

This simulation showed a performance accuracy of 96%. The distributions of this particular data and time point combination appear to be close to the regression line indicating the model's performance.

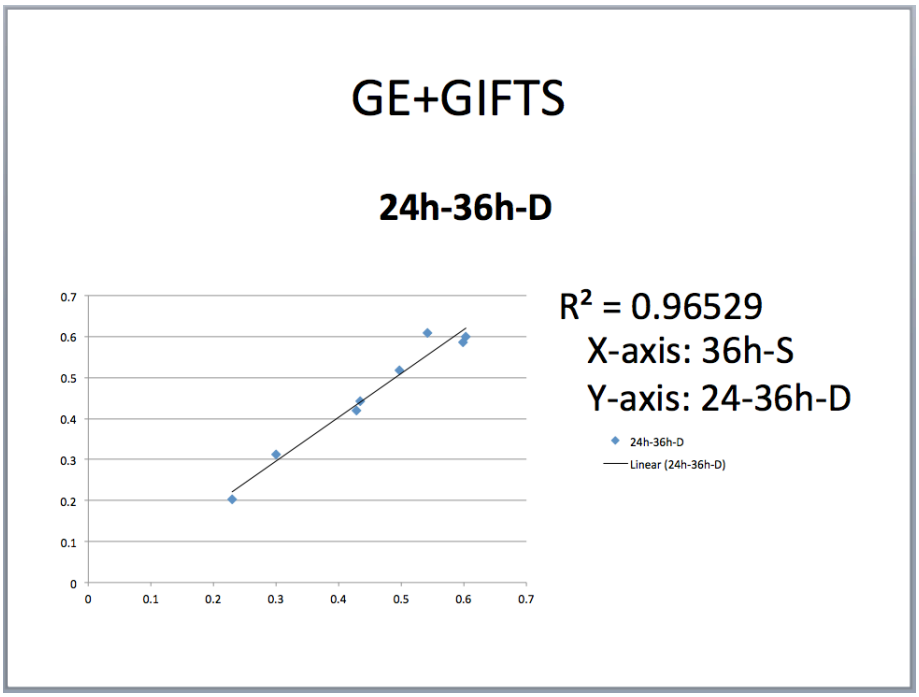

Figure 4.20: Dot plot of 24-hour to 36-hour simulation of GE and GIFtS score combination.

4.1.4 Gene expression data along with both GCI and GIfTS

Interesting things regarding this particular simulation is that the 24-hour to 36-hour simulation, which recorded poor performance in the prior cases, overpassed the other two simulations in this combination with 98%.

4.1.4.1 12-hour to 24-hour simulation

This simulation acted as an initiative to our second experiment as the 12-hour to 24-hour simulation recorded only 57% when all the three data parameters are considered.

Certain edge changes were noted during the simulations and the distribution of data has been presented in the line plots.

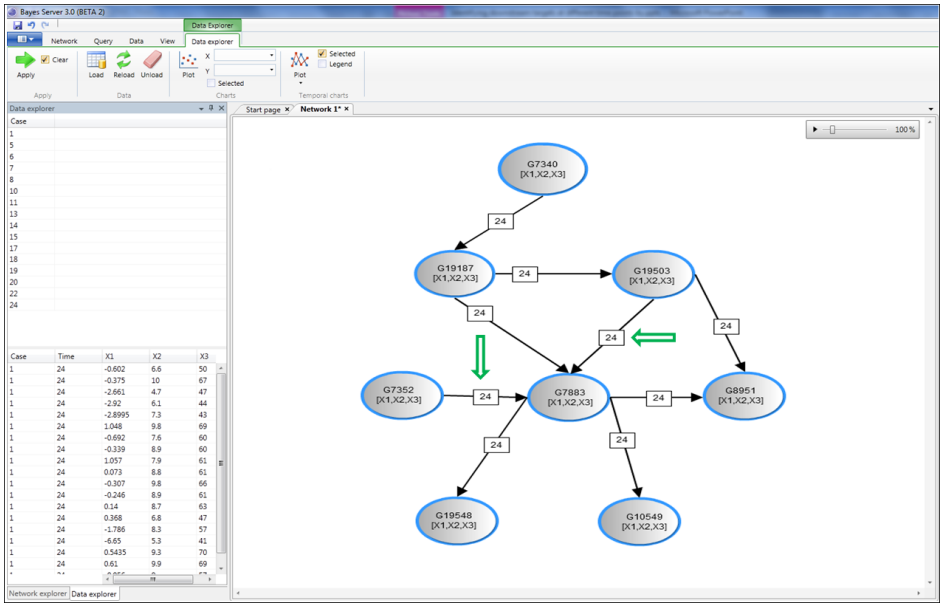

Figure 4.21: 12-hour to 24-hour simulation of all three score combined simulation. Green arrows indicate the edge directionality reversal.

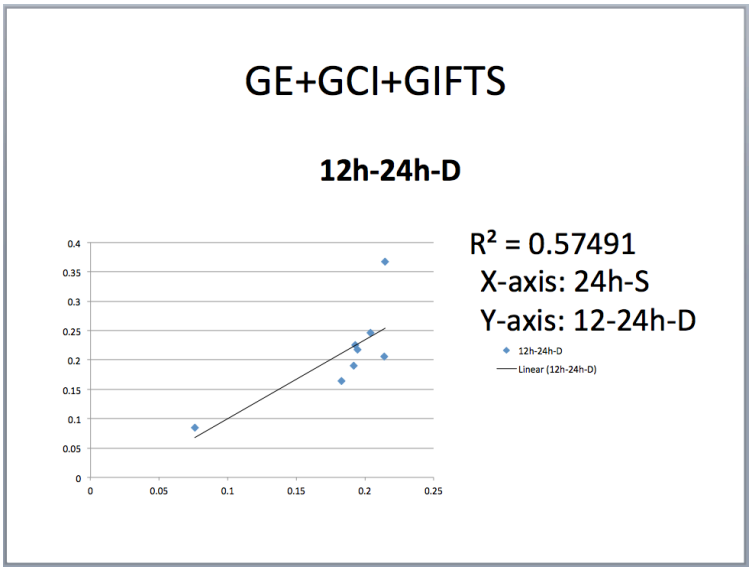

Figure 4.22: Dot plot of 12-hour to 24-hour simulation of all three scores combined

4.1.4.2 12-hour to 36-hour simulation

This simulation yielded 97% accuracy with edge changes in the network produced after the simulation. The figures 4.23 represent the changes below. One important thing to notice here is the total loss of edges between certain nodes and they are left isolated indicating that their interaction is not involved at a certain time point. The violet color arrow indicates the presence of the same edge in the previous simulation

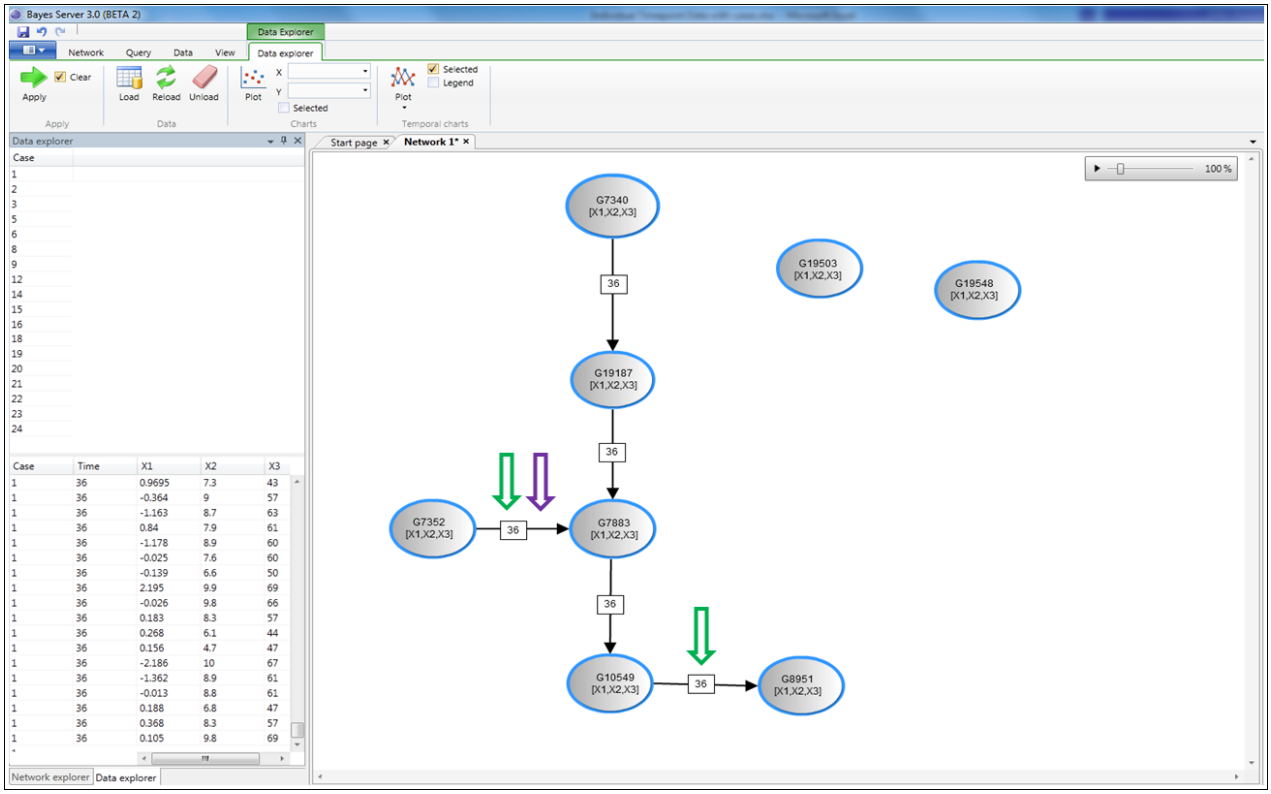

Figure 4.23: 12-hour to 36-hour simulation of all three scores combined together. Isolated nodes were resulted after the simulation.

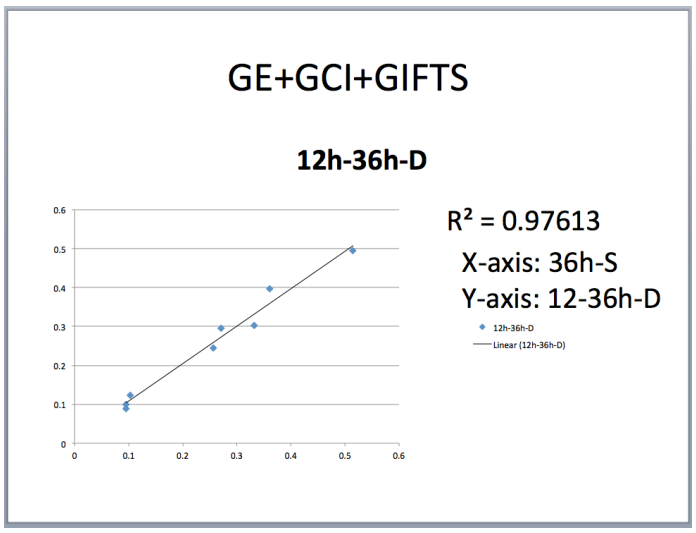

Figure 4.24: Dot plot of 12-hour to 36-hour all three scores combined simulation.

4.1.4.3 24-hour to 36-hour simulation

This simulation stood top in this time and data parameter combination with 98% prediction performance. The two violet colored arrows indicate the presence of same edges in the previous simulation.

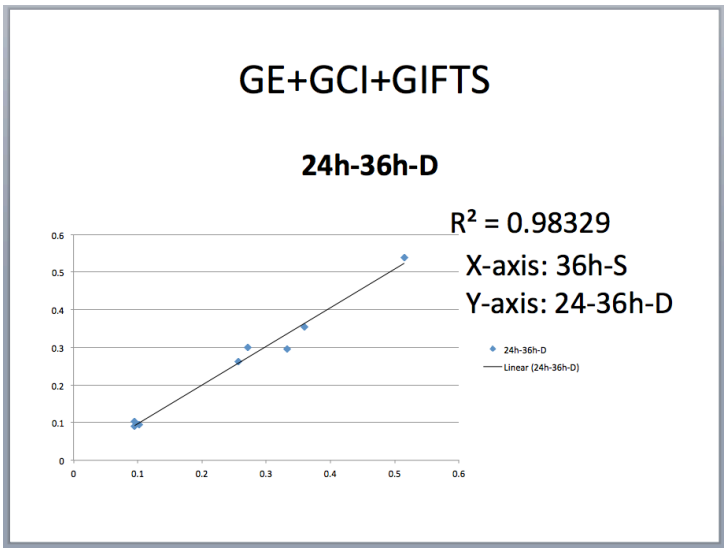

Figure 4.25: Dot plot of 24-hour to 36-hour simulation of all three scores combined.

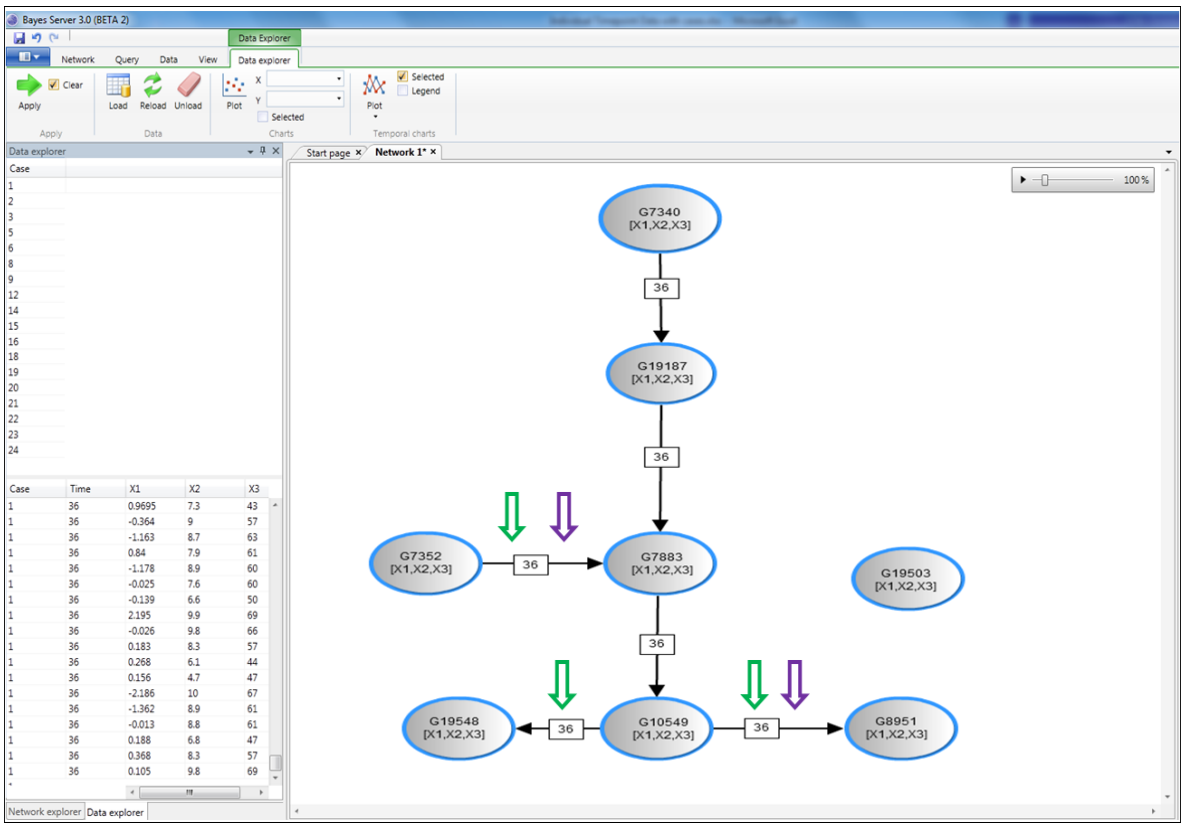

Figure 4.26: 24-hour to 36-hour simulation when all the three scores are combined together. Violet color arrows indicate the presence of such edges in the previous simulations and the green arrows indicate the edge directionality reversal.

4.1.5 Summary of the 8-geneset simulations at 500 parameter-learning iterations

From all the simulations, it is evident that gene expression data's 12-hour to 24-hour data simulation has crossed the other simulation performances of the dynamic Bayesian models (Table 4.4). To understand more in detail about the DBN performance, we employ the second experiment with more number of genes and varying iteration count.

Data Combination	Time points	R^2 for 8 genes at 500 iterations
GE	12-24	0.994
	12-36	0.971
	24-36	0.914
GE+GCI	12-24	0.991
	12-36	0.951
	24-36	0.972
GE+GIFTS	12-24	0.992
	12-36	0.986
	24-36	0.965
GE+GCI+GIFTS	12-24	0.574
	12-36	0.976
	24-36	0.983

Table 4.4: Summary of prediction performance accuracies of the DBN model for the 8-geneset at 500 parameter-learning iterations.

4.2 Experiment 2

The simulation was repeated by constructing the 12-hour static Bayesian network for the 19 genes identified by reducing the correlation coefficient from 0.93 to 0.90, followed by consideration of the 12 different combinations and declaring the node evidences for each gene in the network. The earlier 8 nodes have also resulted in addition to another 11 nodes.

4.2.1 19-gene set simulation at 300 iterations

As usual, the 12-hour static Bayesian network is constructed with the expression data loaded and the simulation was started. The lowest accuracy of the DBN model was for the 12-hour to 36-hour simulation for the expression data-GCI score combination and expression data with GIFtS score, which was only 44%. This shows that when the node

count and parameters increases, it is necessary to let the model learn the parameters for sufficient number of times. The highest was 65% for the traditional only gene expression data simulation at 12-hour to 24-hour. The accuracies of all the different data and time point combinations are mentioned in Table 4.5.

4.2.2 19-gene set simulation at 500 iterations

4.2.2.1 Only with gene expression data

This section explains the simulations carried out using expression.

4.2.2.1.1 12-hour to 24-hour simulation

This simulation resulted in a prediction accuracy of 73%. Since only one parameter was involved, it showed an average prediction performance. The dot plot of this simulation is presented in the figure 4.27.

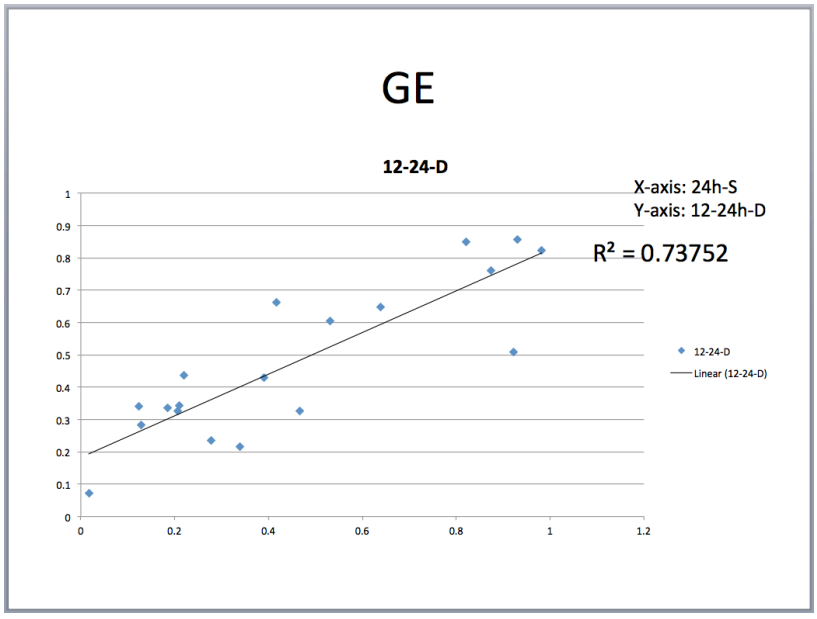

Figure 4.27: Dot plot of 19-geneset's 12-hour to 24-hour simulation at 500 iterations.

4.2.2.1.2 12-hour to 36-hour simulation

This simulation showed accuracy of around 66%. The data distribution is represented in the dot plot in figure 4.28.

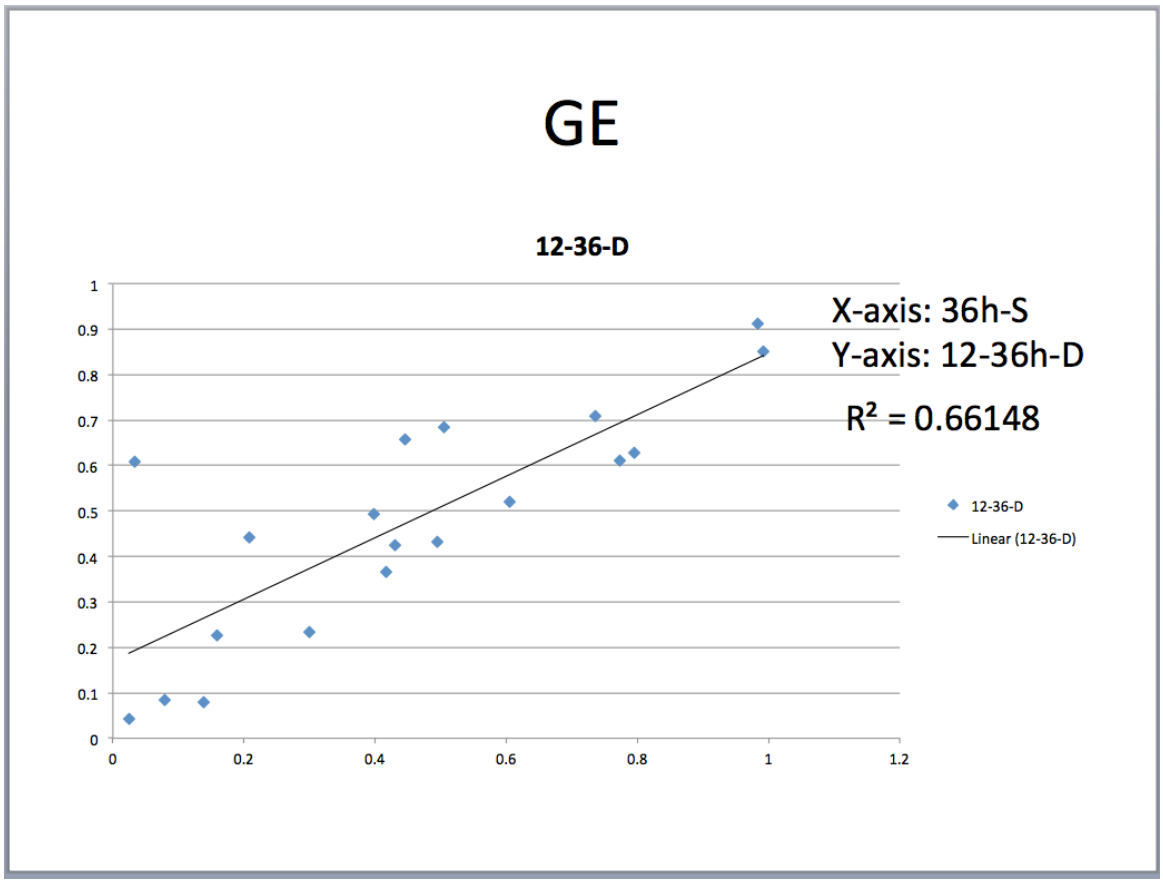

Figure 4.28: Dot plot of 19-geneset's 12-hour to 36-hour simulation at 500 iterations.

4.2.2.1.3 24-hour to 36-hour simulation

This simulation showed an accuracy of 84%. The prediction's distributions are displayed in figure 4.29.

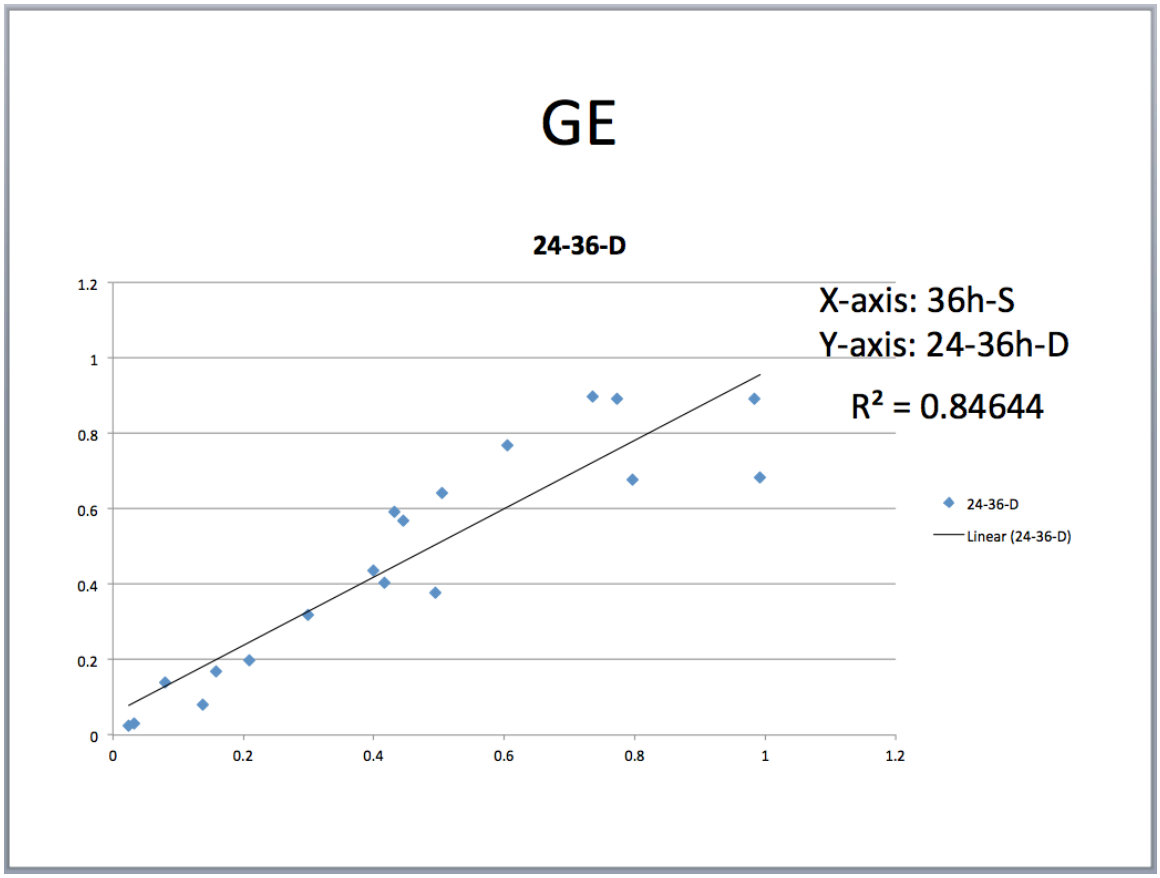

Figure 4.29: Dot plot of 24-hour to 36-hour 19-geneset simulations at 500 iterations.

4.2.2.2 Gene expression data with Gene Characterization index score

This section explains the simulations carried out using expression data and the GCI score.

4.2.2.2.1 12-hour to 24-hour simulation

This simulation yielded a prediction performance of 68%. The dot plot indicates the model's node CPD's distributions in figure 4.30.

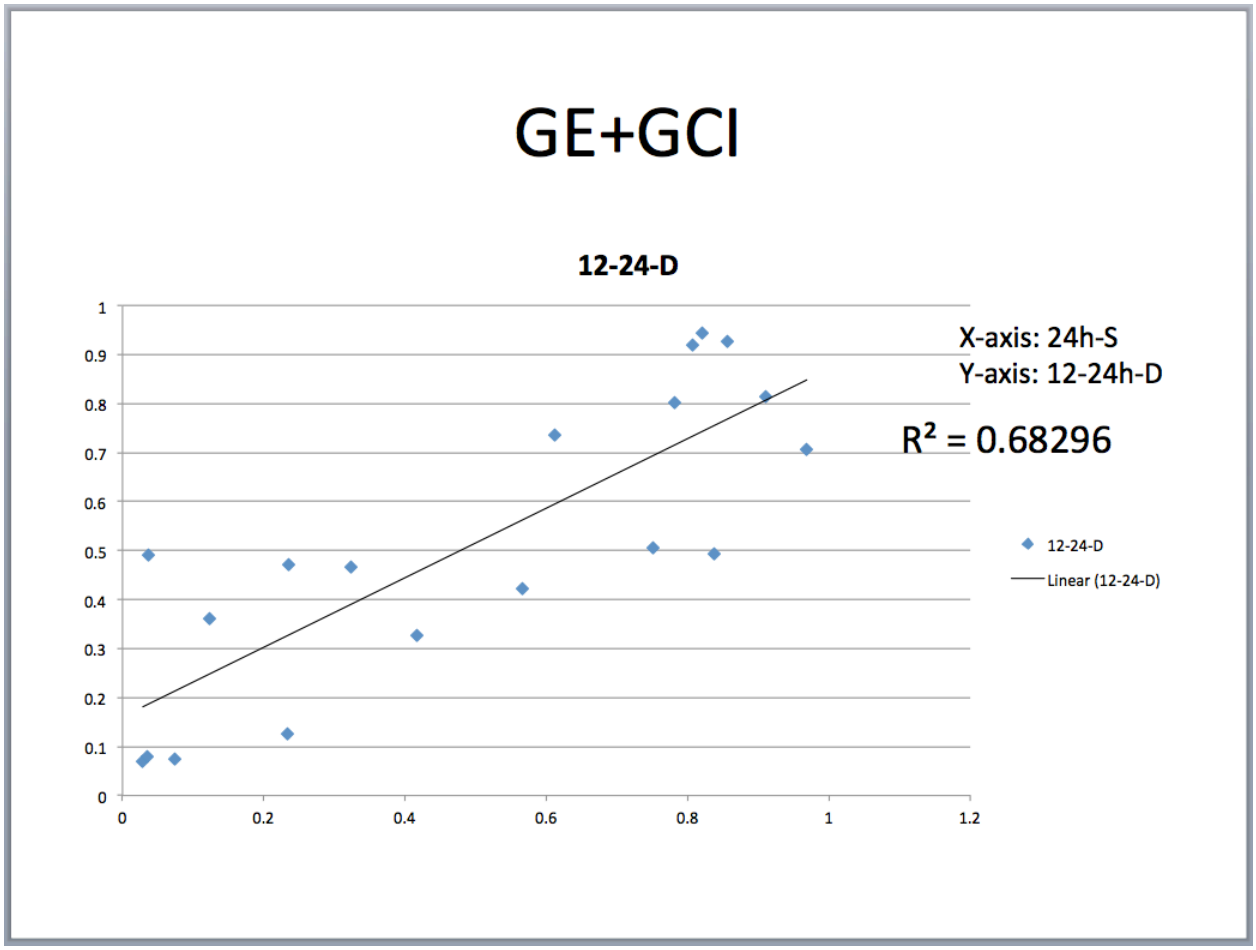

Figure 4.30: Dot plot of node CPDs of 12-hour to 24-hour simulation of 19-geneset at 500 iterations with gene expression data and GCI score combined.

4.2.2.2.2 12-hour to 36-hour simulation

This simulation yielded an accuracy of 53%, which is the second, least performance among the set of simulations performed in this category. Figure 4.31 indicates the node CPD distributions.

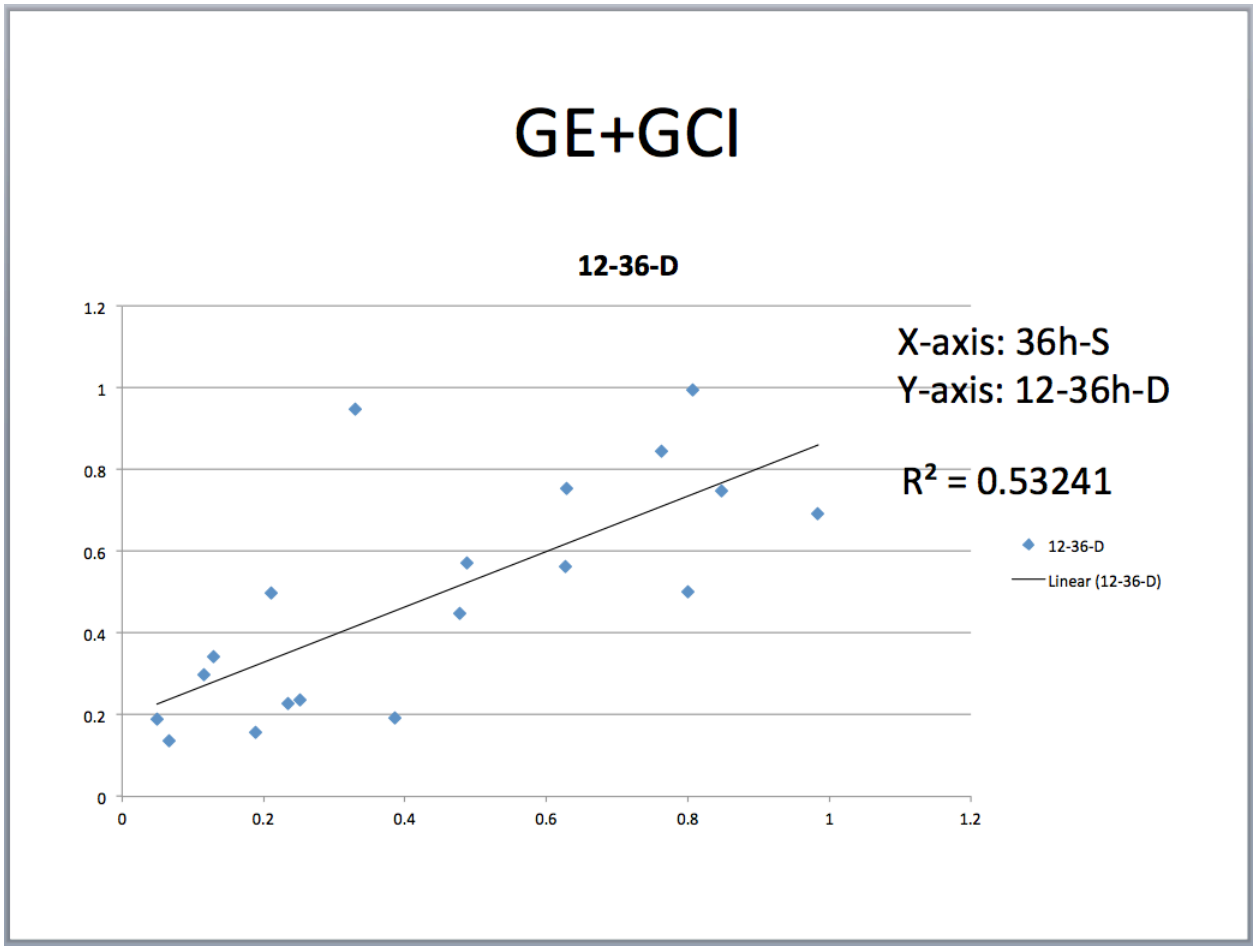

Figure 4.31: Dot plot of node CPDs of 12-hour to 36-hour simulation of gene expression data and GCI score at 500 iterations for the 19-geneset.

4.2.2.2.3 24-hour to 36-hour simulation

This simulation's accuracy was recorded to be 57%, slightly above the average accuracy of the total performance of the model at 500 iterations. This simulation follows the 12-hour to 24-hour simulation, which recorded an accuracy of 68%. The CPDs distribution is presented in figure 4.32.

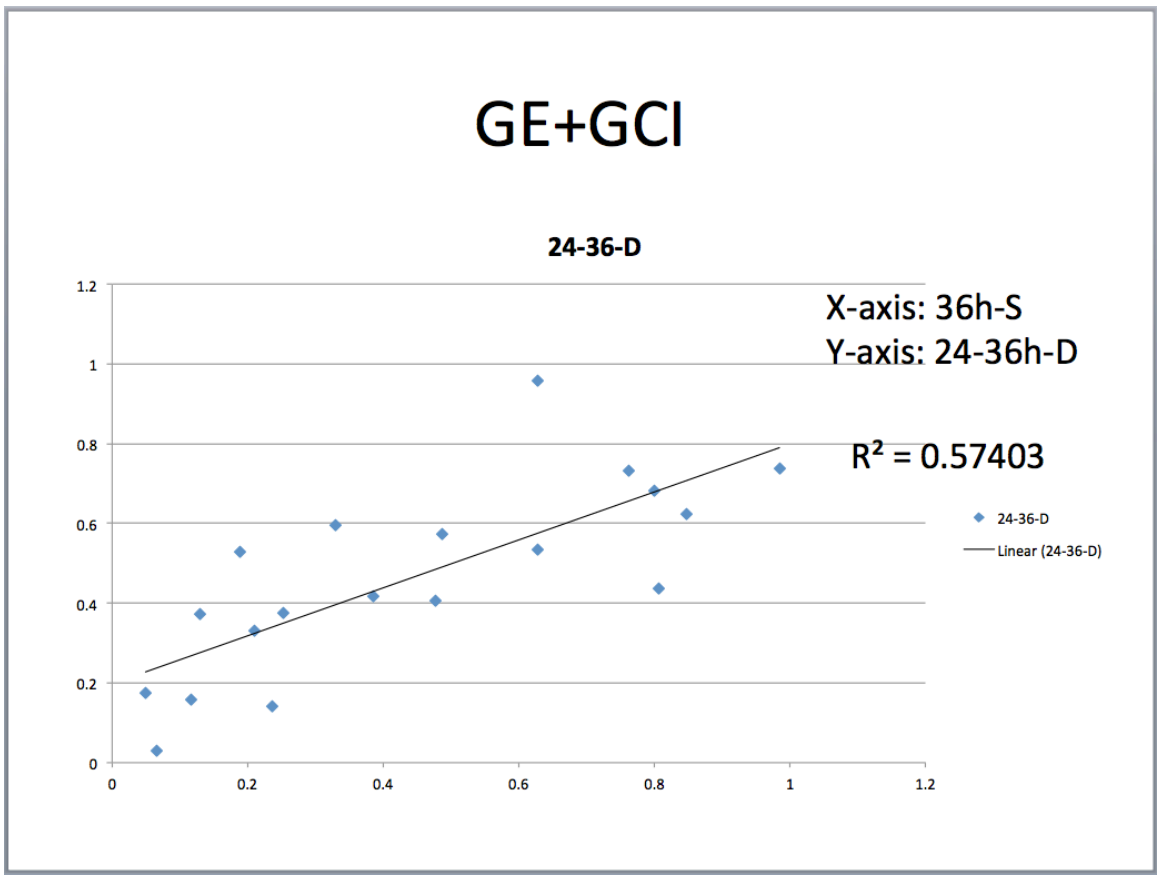

Figure 4.32: Dot plot of CPDs of 24-hour to 36-hour simulation of 19-geneset at 500 iterations.

4.2.2.3 Gene expression data with GIFtS score

This section explains the simulations carried out using expression data and the GIFtS score.

4.2.2.3.1 12-hour to 24-hour simulation

This simulation's performance was observed to be 60%, which was top among the different time point simulation in this section. Figure 4.33 explains the node CPD distributions.

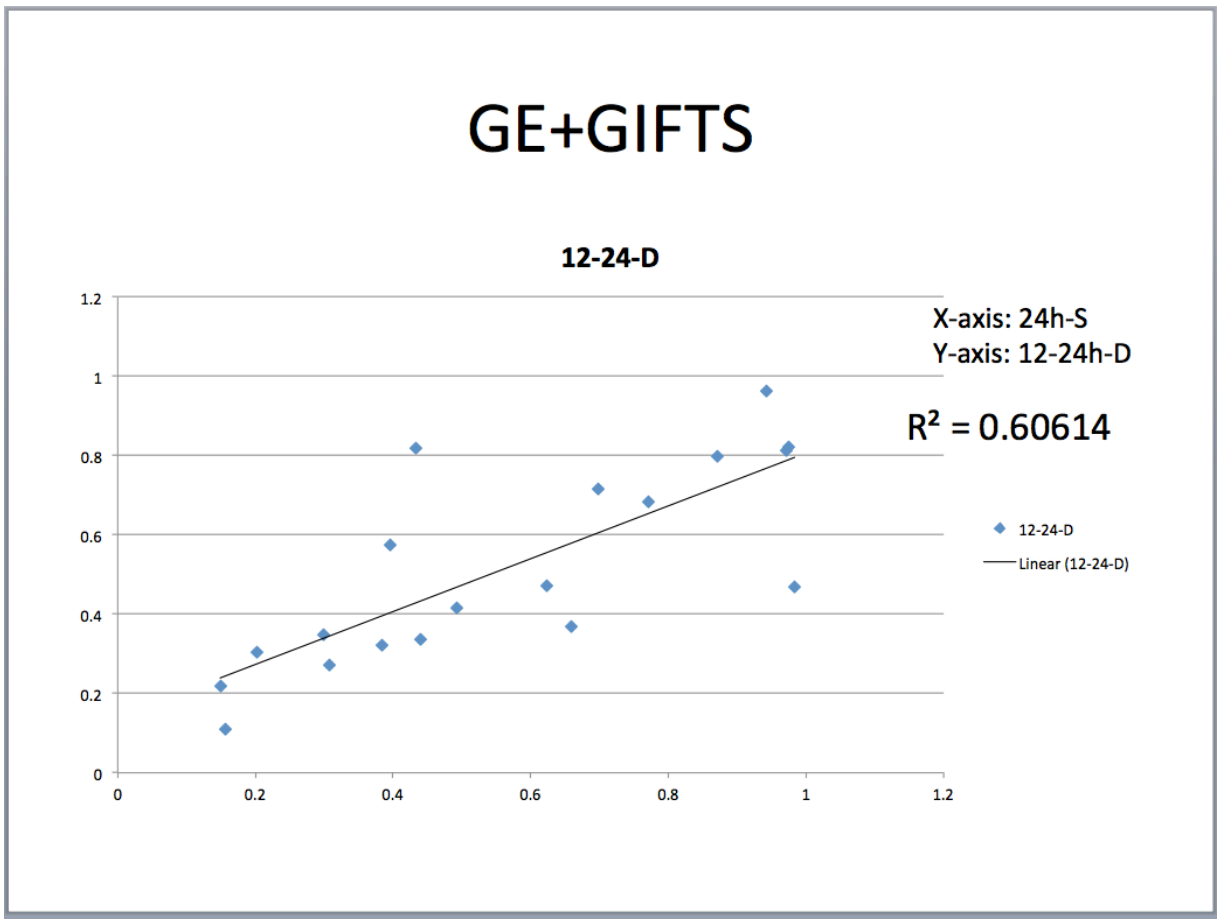

Figure 4.33: Node CPD distributions of 12-hour to 24-hour simulation of gene expression and GIFtS score.

4.2.2.3.2 12-hour to 36-hour simulation

This simulation resulted in a 50% prediction, which was the least for this data combination at 500 iterations. The prediction percentage of this simulation was the least in all the 500 iterations for 19-geneset. The reason might be the passage of information at different time-point levels, i.e. instead of either from 12-hour to 24-hour or 24-hour to 36-hour, the information is passing from 12-hour to 36-hour. The node CPDs are represented in figure 4.34.

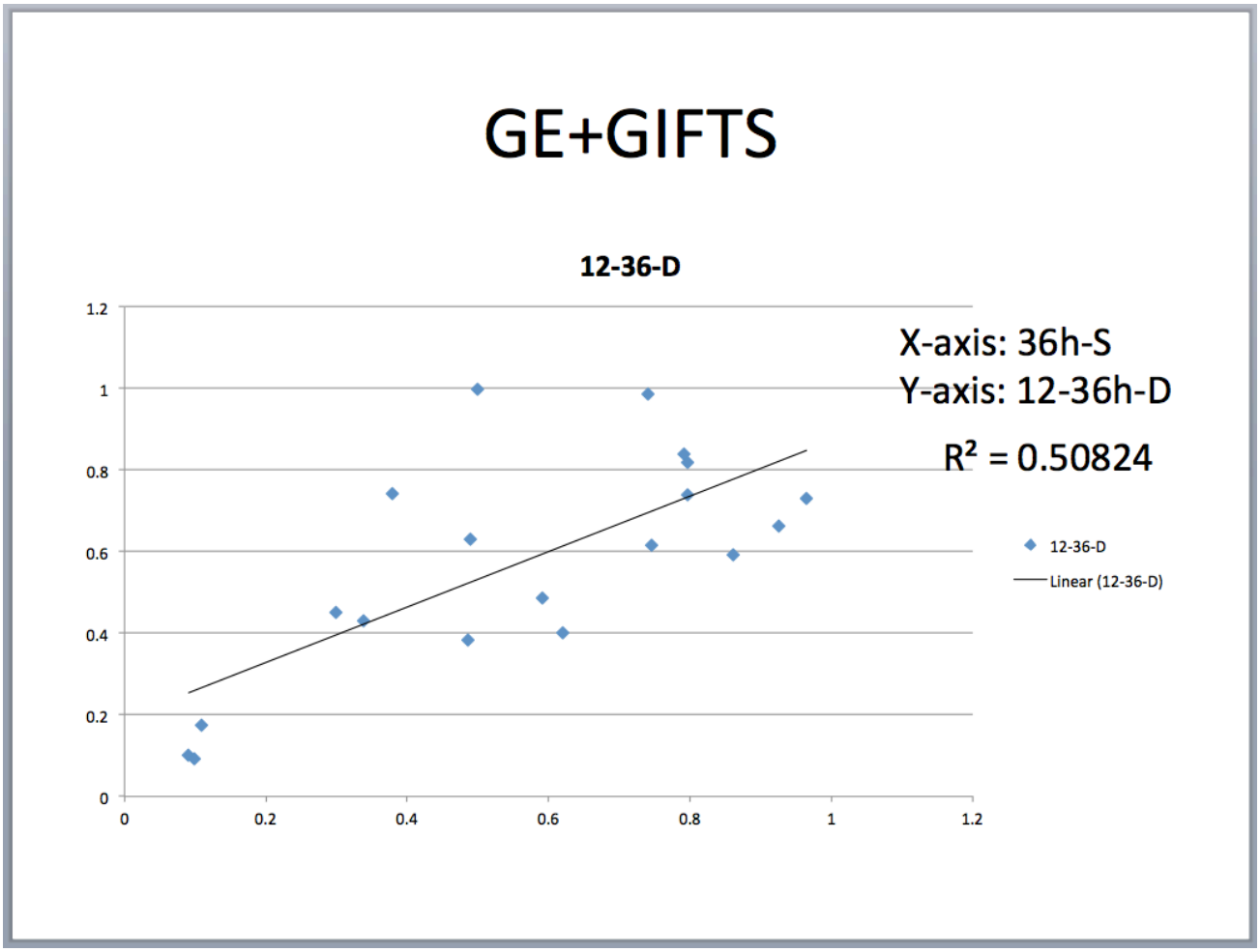

Figure 4.34: Node CPDs plotted using dot plot for the gene expression and GIFtS score combinations for the 19-genest at 500 iterations.

4.2.2.3.3 24-hour to 36-hour simulation

This simulation showed a prediction performance of 56%, the third lowest in the entire simulation section. Compared to the previous 300 iterations, the prediction performance was recorded to be 49% before and has increased by 7% in this iteration. Figure 4.35 explains the node CPDs coverage on the dot plot.

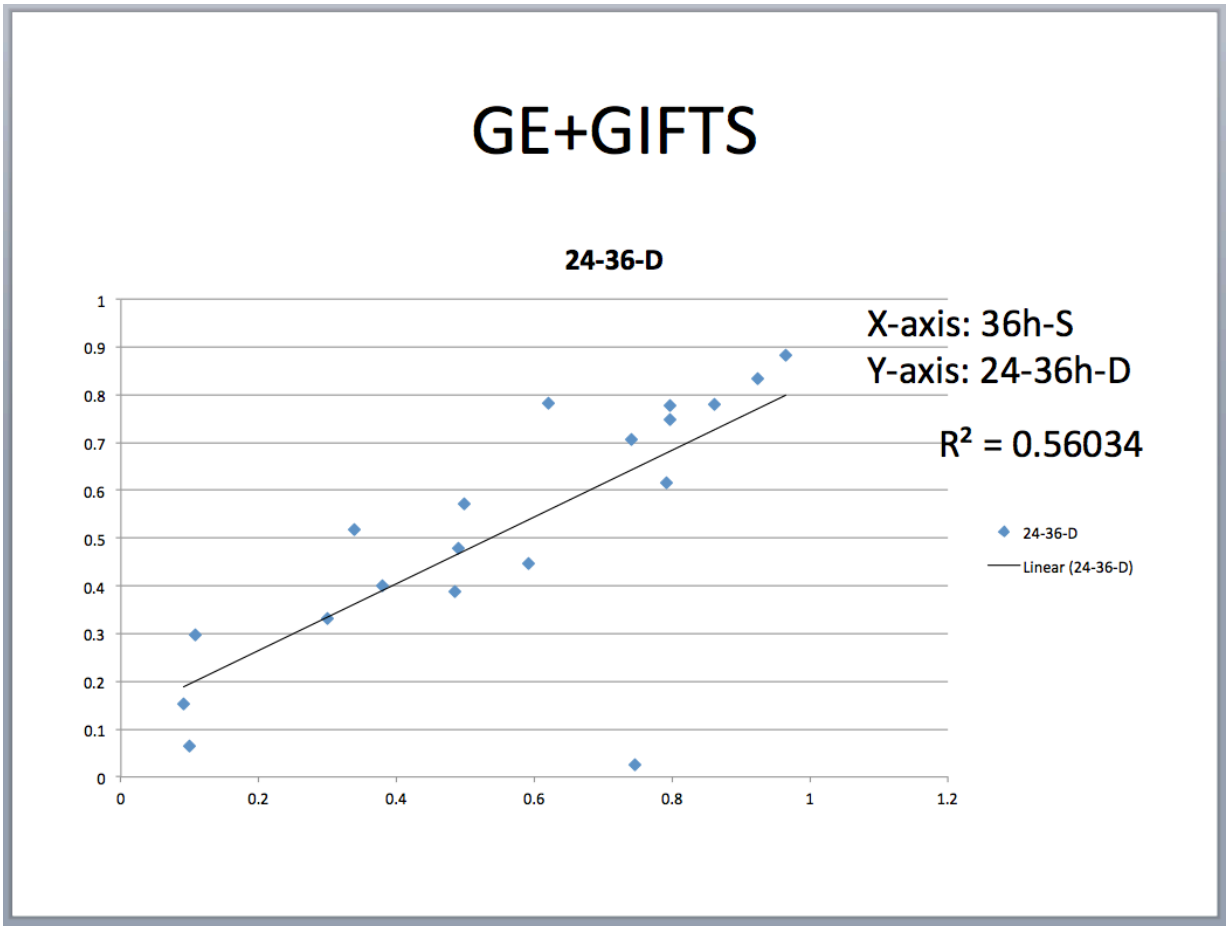

Figure 4.35: Node CPD distribution of 24-hour to 36-hour simulation of the 19-geneset when gene expression and GIFtS are combined together and simulated at 500 iterations.

4.2.2.4 Gene expression data with both GCI and GIFtS scores

This section tries to understand the prediction performances of the DBN model when all the three parameters are combined at 500 iterations for a gene set of 19 genes.

4.2.2.4.1 12-hour to 24-hour simulation

The simulation accuracy was identified to be 56% and that was the second highest among the different time point combinations with 19-genes at 500 iterations. Figure 4.36 indicates the node CPD distributions.

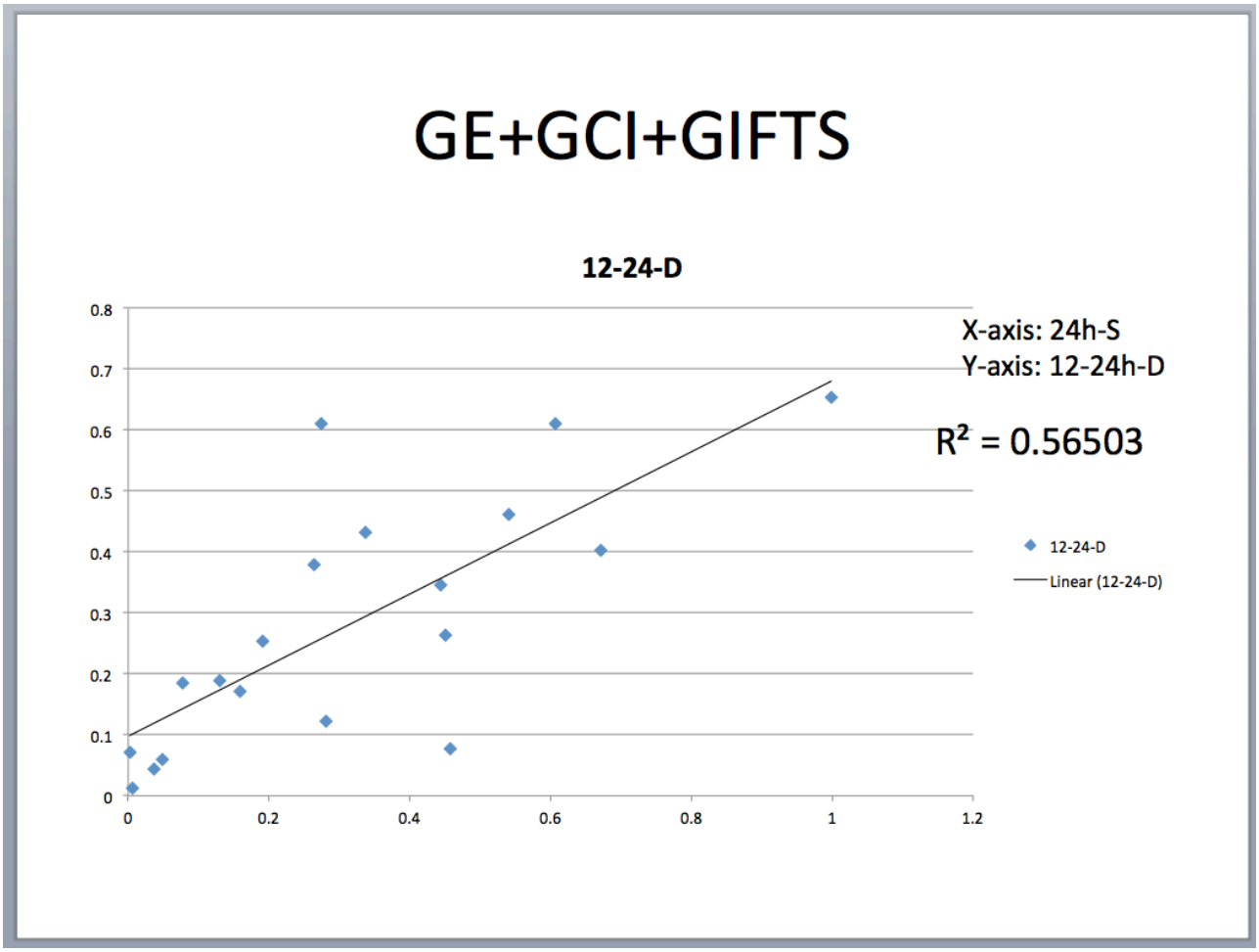

Figure 4.36: DBN model's performance while simulation 12-hour data into 24-hour for all the three data parameter combinations.

4.2.2.4.2 12-hour to 36-hour simulation

The simulation accuracy was identified to be 54%, which was the least of all the three time point simulations carried out at this data combination with 19-genes at 500 iterations. Figure 4.37 represents the model's prediction performance.

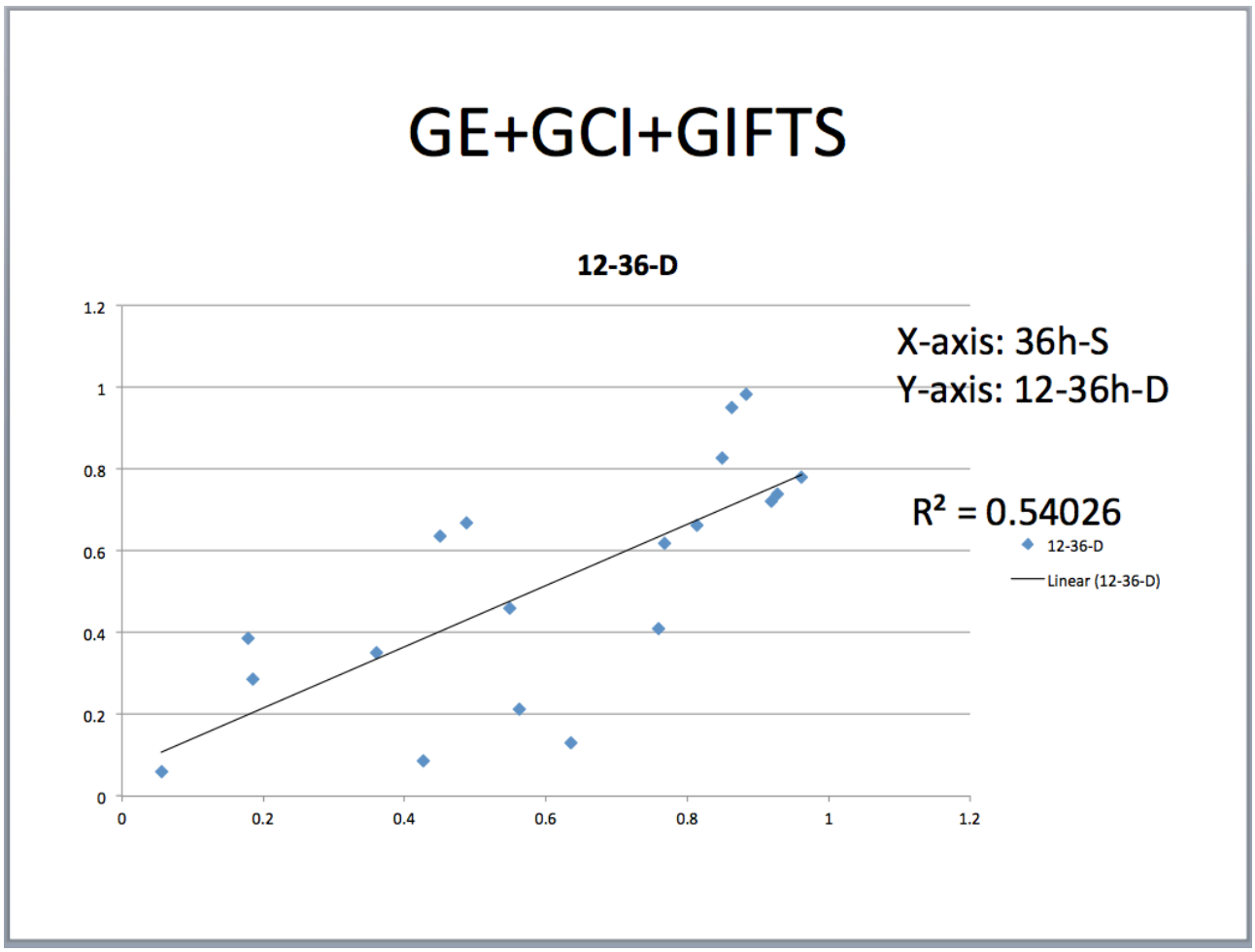

Figure 4.37: DBN model's performance while simulation 12-hour data into 36-hour for all the three data parameter combinations.

4.2.2.4.3 24-hour to 36-hour simulation

The simulation accuracy was identified to be 79%, the highest among all the 500 iterations' data and time point combinations. The reason is that the presence of all the three parameters and ample amounts of learning time with a less noisy data. Figure 4.38 explains the model's performance.

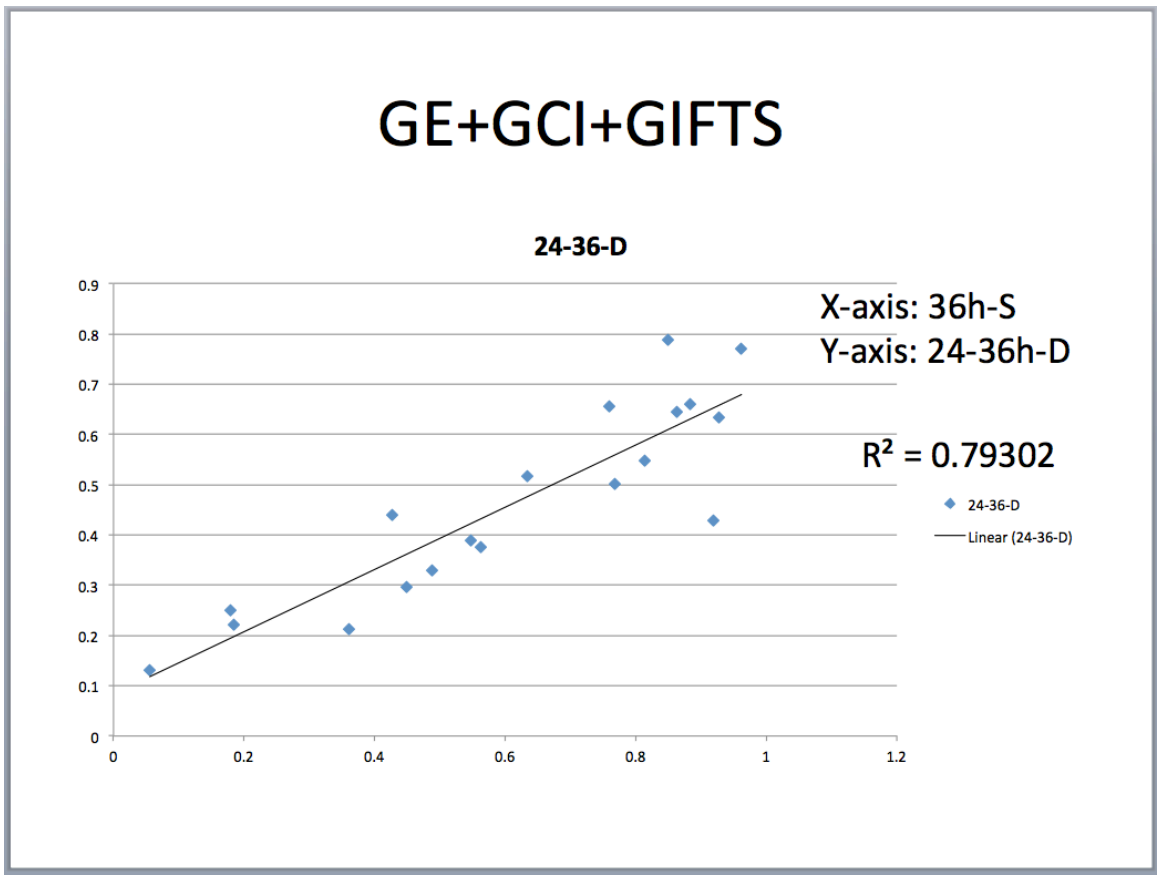

Figure 4.38: DBN model's performance while simulation 24-hour data into 36-hour for all the three data parameter combinations.

4.2.3 19-gene set simulation at 700 iterations

The highest accuracy obtained in this particular section of simulations was 85% for the simulations carried out with only gene expression data at 24-hour to 36-hour simulation. Gene expression in association with GCI, showed a better prediction accuracy of 78%, followed by the gene expression's combination with GIFtS, both for 12-hour to 24-hour simulation (Table 4.5). The lowest accuracy of the DBN model was for the 12-hour to 24-hour simulation for the expression-GCI-GIFtS score combination, which was 64%. This again stresses the fact that when the node count and parameters increases, it is

necessary to let the model learn the parameters for sufficient number of times. Basically, this practice of gradually increasing the parameter learning count enables an individual to assess the model's performance in a stage-by-stage method. The accuracies in this section compared against the ones at 300 and 500 iterations were less because of the learning time the model has been awarded and we expected that the next and last stage of iteration count, 1000, will definitely results a satisfactory accuracy rate.

4.2.4 19-gene set simulation at 1000 iterations

4.2.4.1 Only gene expression data

This simulation is carried out using the expression data only. At an average, the model's performance accuracy was around 88%. The 24-hour to 36-hour simulation ranked highest in terms of accuracy.

4.2.4.1.1 12-hour to 24-hour simulation

With an accuracy of 85%, the gene expression data simulation ranked the least among the three different time point simulation. Figure 4.39 represents the distribution of data.

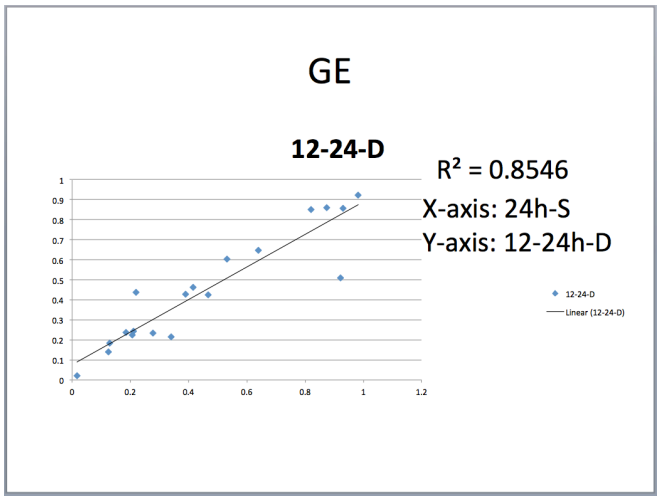

Figure 4.39: 12-hour to 24-hour simulation of 19-genes at 1000 iterations

4.2.4.1.2 12-hour to 36-hour simulation

This simulation is the second highest in terms of DBN's prediction accuracy of 88%. Figure 4.40 represents the node CPDs distribution.

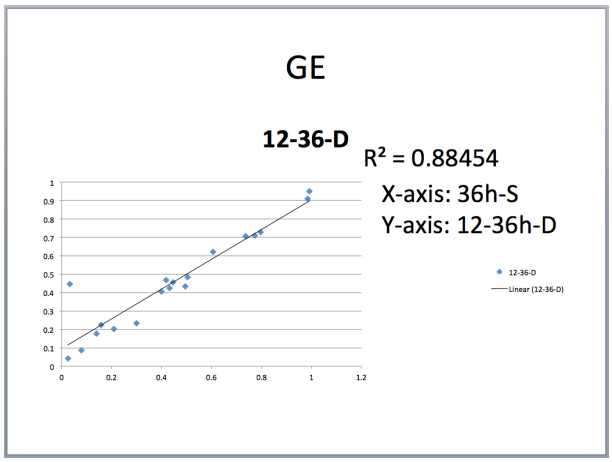

Figure 4.40: 12-hour to 36-hour simulation of 19-genes at 1000 iterations

4.2.4.1.3 24-hour to 36-hour simulation

The 24-hour to 36-hour simulation ranked highest in connection to the accuracy of the prediction of one time point from another, with 93%, which is the highest of all the simulations at 1000 iterations. Figure 4.41 represents the dot plot of node CPD distributions.

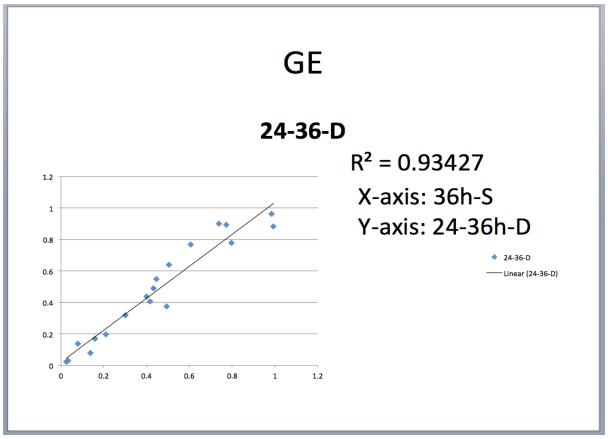

Figure 4.41: 24-hour to 36-hour simulation of 19-genes at 1000 iterations

4.2.4.2 Gene expression with GCI score

These simulations are carried out using the combination of gene expression and GCI score. Similar to the earlier data simulation, the gene expression data simulation, this simulation also showed a high prediction accuracy of the 24-hour to 36-hour simulation.

4.2.4.2.1 12-hour to 24-hour simulation

83% accuracy was observed when the model simulated 24-hour time point data from 12-hour time point data. Figure 4.42 shows the dot plots of prediction.

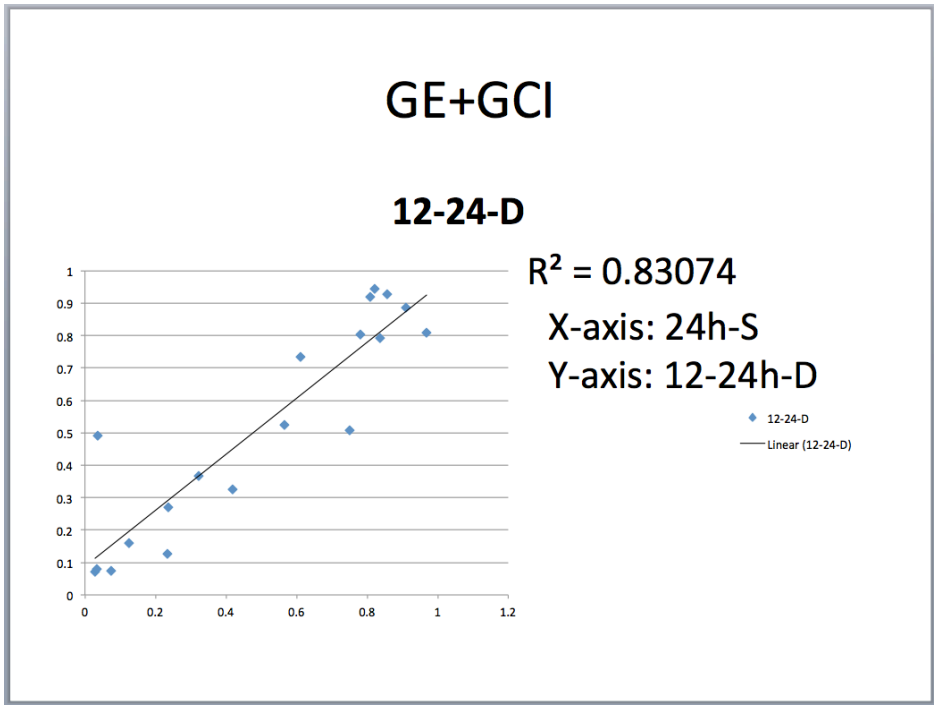

Figure 4.42: 12-hour to 24-hour simulation with GE and GCI score together

4.2.4.2.2 12-hour to 36-hour simulation

The simulation's accuracy of 83% is the second highest in this time and data point combination. Figure 4.43 represents the accuracy and CPD distributions in a dot plot.

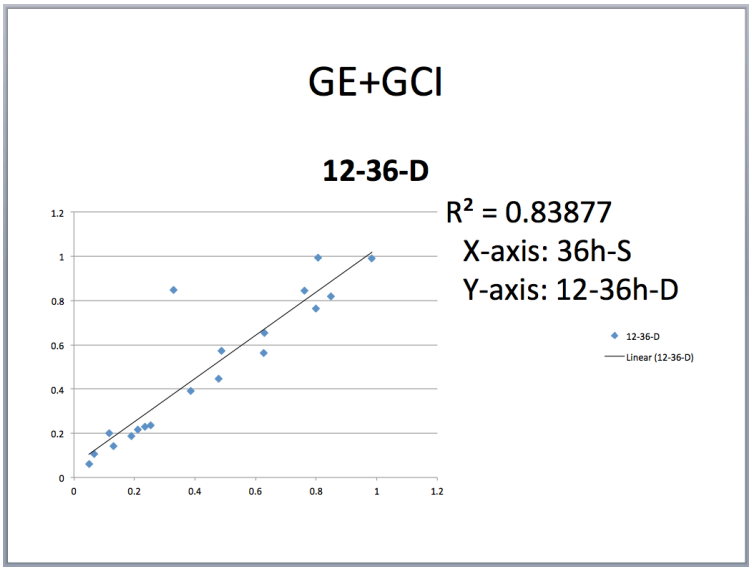

Figure 4.43: 12-hour to 36-hour simulation with GE and GCI score together

4.2.4.2.3 24-hour to 36-hour simulation

It was observed that the accuracy of this simulation was 92%, placing this data and time point combination on the top. Figure 4.44 represents the dot plot.

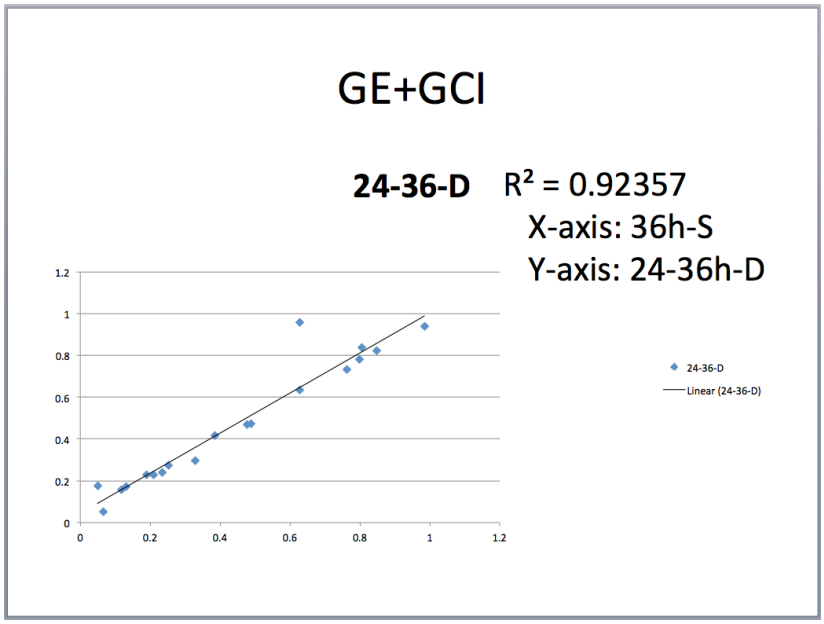

Figure 4.44: 24-hour to 36-hour simulation with GE and GCI score together

4.2.4.3 Gene expression with GIFtS score

The gene expression data was combined with the GIFtS score for the 19-geneset and the simulations were carried out at 1000 parameter learning iterations. The average accuracy of this particular simulation section was 90%.

4.2.4.3.1 12-hour to 24-hour simulation

The accuracy of this simulation was 91%; second highest in the data and time point combination. Figure 4.45 represents the dot plot of CPD distribution.

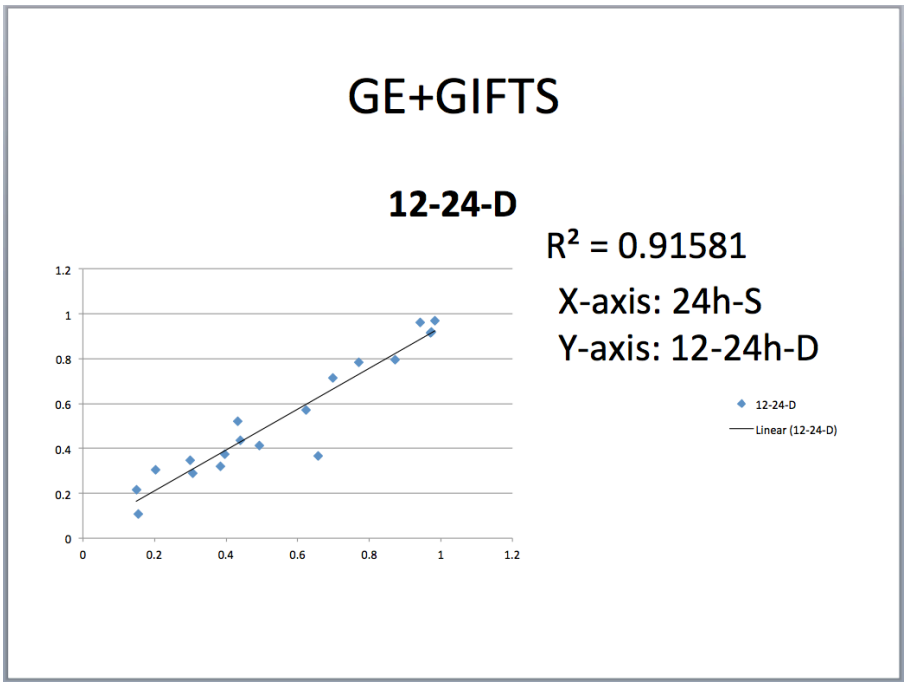

Figure 4.45: 12-hour to 24-hour simulation with GE and GIFtS score together

4.2.4.3.2 12-hour to 36-hour simulation

The accuracy was 92%, the highest in this section and also the second highest of all the simulations carried out at 1000 iterations. Figure 4.46 explains the node CPD distributions.

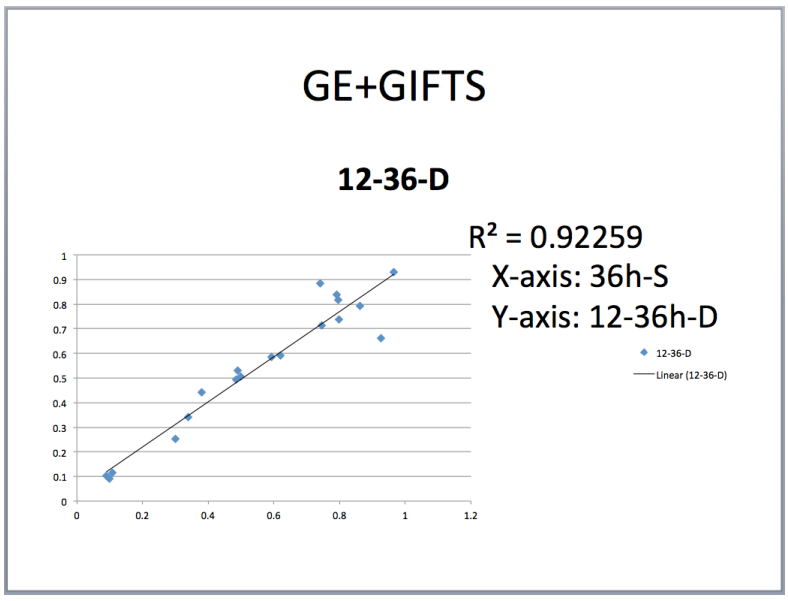

Figure 4.46: 12-hour to 36-hour simulation with GE and GIFtS score together

4.2.4.3.3 24-hour to 36-hour simulation

With 90% of accuracy, this simulation is the least of all the three time point simulation in this category. Figure 4.47 gives an overview of the distribution using a dot plot.

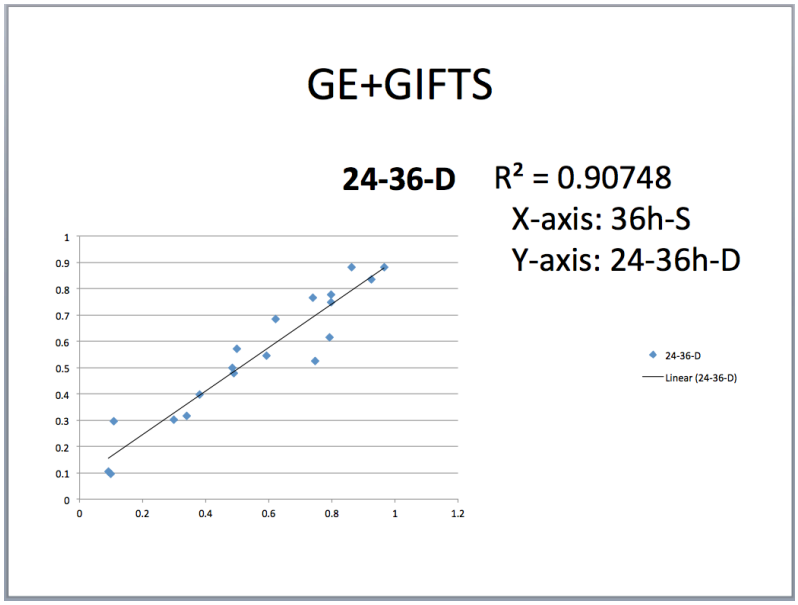

Figure 4.47: 24-hour to 36-hour simulation with GE and GIFtS score together

4.2.4.4 Gene expression with GCI and GIFtS score

All the three score, gene expression, GCI and GIFtS are combined together for this section of simulations. Table 4.5 in this section explains the summary of correlation values of the 8-genes and 19-genes' at their respective number or iterations.

4.2.4.4.1 12-hour to 24-hour simulation

The least accuracy in this section was for this simulation, which was 80%. Figure 4.48 explains the node CPD distributions.

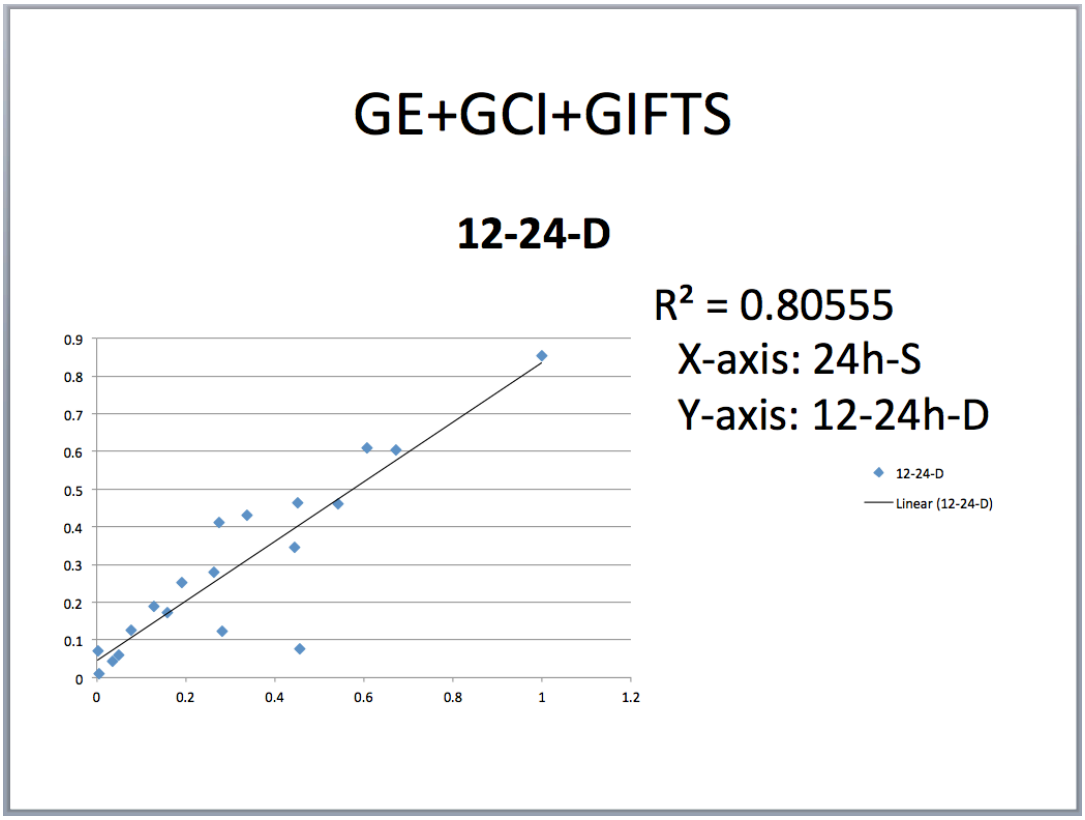

Figure 4.48: 12-hour to 24-hour simulation with all three scores together

4.2.4.4.2 12-hour to 36-hour simulation

This simulation was predicted with an accuracy of 87%, the second highest in this data and time point combination. Figure 4.49 explains the node CPD distributions.

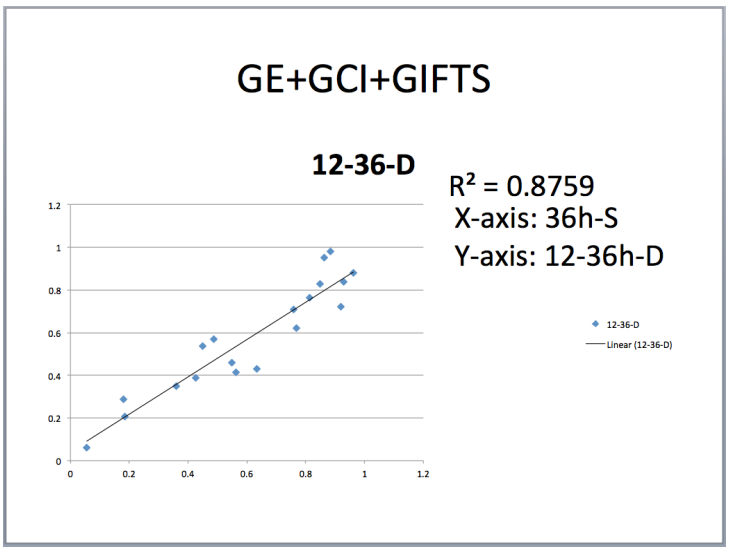

Figure 4.49: 12-hour to 36-hour simulation with all three scores together

4.2.4.4.3 24-hour to 36-hour simulation

The last but not the least simulation to 36-hour from 24-hour time point data resulted in an accuracy of 88%, the highest among this section's simulations. Figure 4.50 indicates the dot plot of the 24-hour to 36-hour simulation at 1000 iterations with the 19-geneset.

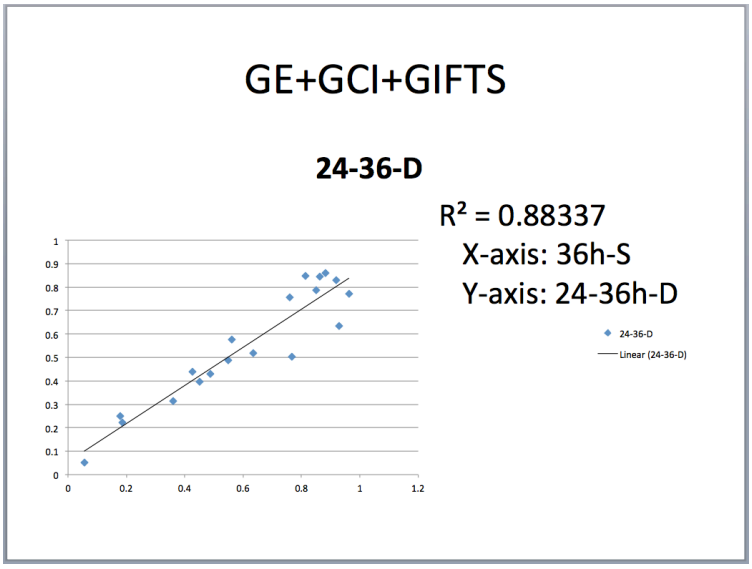

Figure 4.50: 24-hour to 36-hour simulation of 19 genes with all three scores together

Data Combination	Time points	R^2 for 8 genes	R^2 for 19 genes			
		500 iterations	300 iterations	500 iterations	700 iterations	1000 iterations
GE	12-24	0.994	0.656	0.737	0.751	0.854
	12-36	0.971	0.569	0.661	0.710	0.884
	24-36	0.914	0.749	0.846	0.853	0.934
GE+GCI	12-24	0.991	0.604	0.683	0.787	0.830
	12-36	0.951	0.440	0.532	0.711	0.838
	24-36	0.972	0.451	0.574	0.774	0.923
GE+GIFTS	12-24	0.992	0.560	0.606	0.755	0.915
	12-36	0.986	0.440	0.508	0.750	0.922
	24-36	0.965	0.499	0.560	0.739	0.907
GE+GCI+GIFTS	12-24	0.574	0.508	0.565	0.644	0.805
	12-36	0.976	0.494	0.540	0.701	0.875
	24-36	0.983	0.608	0.793	0.801	0.883

Table 4.5: Summary of all the 8-genes at 500 iterations and 19-genes at 300, 500, 700 and 1000 iterations. The correlation values for each data and time point combinations are mentioned in the table.

5. DISCUSSION

Initially, when the iteration count was set at 300 and simulations were carried out, it resulted in a very low r-value for a majority of the data combinations in each simulation. In order to identify the best iteration count and the point at which the model's predictions reaches a certain level of expected efficacy, simulations were carried out at 500, 700 and 1000 iterations. For the three score combination at 12-24h simulation, the model's prediction was relatively less than that of 300 and 500 iterations for the 19-gene set compared to the 8-gene set simulation at 500 iterations. Even though the 700 iteration yielded a good r-value of 0.64, keeping the other simulations in mind, the iteration count was increased to 1000 for the best possible prediction.

When the iteration count was increased to 1000, satisfactory changes were observed in the r-value and also with the conditional probability distributions at each node. One major point worth mentioning is the change that was observed with the 24-hour to 36-hour simulation only with gene expression data as opposed to the 12-hour to 24-hour simulation with all the three parameters.

The model's efficiency at 1000 iterations was higher in both cases when compared to the 8 and 19 gene simulations with 500 iterations. It is of prime importance that incorporating an additional parameter that accounts for a node in the Bayesian model might yield positive and pertinent information with adequate training. In a situation at more number of nodes in a Bayesian model, this particular practice of incorporating additional knowledge to the nodes will yield the best solution.

In the dynamic Bayesian model, the incorporation of the GCI score, which is an annotation score of the genes involved in the human genome, has helped the model after a sufficient number of iterations in predicting the 24 hour and 36 hour probabilistic distributions, using the 12 hour gene expression data along with the GCI score mix. Even with the GIFtS score and the combination of all the three scores at 1000 iterations showed a satisfactory prediction.

A comparison study between GCI and GIFtS among 489 genes have shown that the GCI score is 10, while the GIFtS score varied from 51 to 84. GCI claims 33,000 genes to be protein coding, whereas, GIFtS suggests that approximately ~22,000 genes are defined to be protein coding on the basis of knowledge from uniprot, RefSeq and ensemble. Also, GIFtS allows the user to search for genes under different categories like expression, pathways and disorders. Thus, GIFtS produces a fine line of division between the genes where GCI fails to do so.

6. CONCLUSION

Adding parameters that are based on the annotation information to the DBN model has proved that they are helpful in predicting the future time course or downstream targets within a large network.

Adding prior knowledge parameters into the Bayesian model will result in an enhanced accuracy [63]. In our study, we considered two different scores that account for the gene's annotation strength that were included in the Bayesian model. A total of 60 different simulation experiments have been carried out for the 8 and 19-gene sets that included gene expression data with a combination of other parameters. The gene expression simulation stood top for the 12-hour to 24-hour simulation with an r-value of 0.994 for the 8-gene set at 500 iterations. The remaining 92% of the simulation experiments recorded a high r-value when gene expression was combined with other parameters.

In the 8-gene set at 500 iteration simulation, gene expression with GIFtS ranked high for the 12-hour to 24- hour simulation and all the three parameters combination was satisfactory for the 24-hour to 36-hour simulation with an r-value of 0.983. In all the 12 simulations at 1000 iterations, the 19-gene set contained an r-value greater than 0.8. When all the three scores were combined, performed to crosscheck the model's prediction capabilities with a larger set of nodes over the 8- gene set simulation experiment. Sufficient number of training played a major role in the model's prediction ability. Keeping this fact in mind, when the training iteration count was increased from 500 to 1000 for the larger dataset, it yielded positive and satisfactory prediction accuracy. From the complete results shown in Table 2, it can be observed that by using gene

expression data along with other parameters, it is possible to improve the efficiency of Bayesian models and understand the future time-course events. Additionally, by providing a good set of node evidences to the genes in the model, the model can reach accuracy above 85% for more than 90% of the experiments in the model. The conclusion is that, incorporating expression data with additional parameters that account for the prior knowledge about the genes in the expression dataset will yield a better prediction model for larger datasets. However, the selection of best parameter that could be incorporated with time-course gene expression data needs further investigation.

7. REFERENCES

Abhik S, Toyoaki T, & Peter W (Using mechanistic Bayesian networks to identify downstream targets of the Sonic Hedgehog pathway. BMC bioinformatics 10.

Astrand M (2001) Normalizing oligonucleotide arrays. Unpublished manuscript.

Beal MJ, Falciani F, Ghahramani Z, Rangel C, & Wild DL (2005) A Bayesian approach to reconstructing genetic regulatory networks with hidden factors. Bioinformatics 21(3):349.

Bhardwaj N & Lu H (2005) Correlation between gene expression profiles and proteinñprotein interactions within and across genomes. Bioinformatics 21(11):2730.

Bilmes J (2000) Dynamic bayesian multinets. (Citeseer).

Bilmes J & Zweig G (2002) The Graphical Models Toolkit: An open source software system for speech and time-series processing. (IEEE), pp IV-3916-IV-3919.

Bolstad BM, Irizarry RA, Astrand M, & Speed TP (2003) A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. Bioinformatics 19(2):185.

Boyen X & Koller D (1998) Tractable inference for complex stochastic processes. (Citeseer).

Buntine W (2002) A guide to the literature on learning probabilistic networks from data. Knowledge and Data Engineering, IEEE Transactions on 8(2):195-210.

Calza S, Valentini D, & Pawitan Y (2008) Normalization of oligonucleotide arrays based on the least-variant set of genes. BMC bioinformatics 9(1):140.

Changwon K, Jung W, Won O, Sue-Nie P, & Kyoung N (Improving gene expression similarity measurement using pathway-based analytic dimension. BMC Genomics 10.

Charles S, Andrew M, & Khashayar R (2007) Dynamic Conditional Random Fields: Factorized Probabilistic Models for Labeling and Segmenting Sequence Data. Journal of Machine Learning Research 8(2007):693-723.

Chen Y & Xu D (2003) Computational analyses of high-throughput protein-protein interaction data. Current Protein and Peptide Science 4(3):159-180.

Cheng J, Greiner R, Kelly J, Bell D, & Liu W (2002) Learning Bayesian networks from data: an information-theory based approach. Artificial Intelligence 137(1-2):43-90.

Cooper GF (1990) The computational complexity of probabilistic inference using Bayesian belief networks. Artificial Intelligence 42(2-3):393-405.

Dean T & Kanazawa K (1988) Probabilistic temporal reasoning. (AAAI).

Djebbari A & Quackenbush J (2008) Seeded Bayesian Networks: constructing genetic networks from microarray data. BMC Systems Biology 2(1):57.

Dobbs F (1996) A scoring system for predicting group A streptococcal throat infection. The British Journal of General Practice 46(409):461.

Dojer N, Gambin A, Mizera A, Wilczy ski B, & Tiuryn J (2006) Applying dynamic Bayesian networks to perturbed gene expression data. BMC bioinformatics 7(1):249.

Druzdzel MJ (1997) Five useful properties of probabilistic knowledge representations from the point of view of intelligent systems. Fundamenta Informaticae 30(3):241-254.

Efroni S, Schaefer CF, & Buetow KH (2007) Identification of key processes underlying cancer phenotypes using biologic pathway analysis. PLoS ONE 2(5):425.

Freno A (2006) Learning the Structure of Bayesian Networks.

Friedman N, Linial M, Nachman I, & Pe'er D (2000) Using bayesian networks to analyze expression data. Journal of Computational Biology 7(3-4):601-620.

Harel A, et al. (2009) GIFtS: annotation landscape analysis with GeneCards. BMC bioinformatics 10(1):348.

Heckerman D (2008) A tutorial on learning with Bayesian networks. Innovations in Bayesian Networks:33-82.

Heckerman D, Geiger D, & Chickering DM (1995) Learning Bayesian networks: The combination of knowledge and statistical data. Machine learning 20(3):197-243.

Helman P, Veroff R, Atlas SR, & Willman C (2004) A bayesian network classification methodology for gene expression data. Journal of Computational Biology 11(4):581-615.

Huang Z, Li J, Su H, Watts GS, & Chen H (2007) Large-scale regulatory network analysis from microarray data: modified Bayesian network learning and association rule mining. Decision Support Systems 43(4):1207-1225.

Hulst J (2006) Modeling physiological processes with dynamic Bayesian networks. Faculty of Electrical Engineering, Mathematics, and Computer Science, University of Pittsburgh.

Husmeier D (2003) Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks. Bioinformatics 19(17):2271.

Jayesh P, Mehmet K, & Ananth G (Functional characterization and topological modularity of molecular interaction networks. BMC bioinformatics 11.

Jordan MI, Ghahramani Z, Jaakkola TS, & Saul LK (1999) An introduction to variational methods for graphical models. Machine learning 37(2):183-233.

Kalman RE (1960) A new approach to linear filtering and prediction problems. Journal of basic Engineering 82(1):35-45.

Kanabar P, Vaske C, Yeang C, Yildiz F, & Stuart J (2009) Inferring disease-related pathways using a probabilistic epistasis model. (World Scientific Pub Co Inc), p 480.

Kanehisa M (2002) The KEGG database. Silico Simul Biol Proc 247:91-103.

Kemmer D, et al. (2008) Gene characterization index: assessing the depth of gene annotation. PLoS ONE 3(1):1440.

Kerman J & Gelman A (2006) Bayesian data analysis using R. R News 6:21ñ24.

Kim S, Imoto S, & Miyano S (2004) Dynamic Bayesian network and nonparametric regression for nonlinear modeling of gene networks from time series gene expression data. Biosystems 75(1-3):57-65.

Ko Y, Zhai C, & Rodriguez-Zas SL (2007) Inference of gene pathways using Gaussian mixture models. (IEEE), pp 362-367.

Ko Y, Zhai CX, & Rodriguez-Zas S (2009) Inference of gene pathways using mixture Bayesian networks. BMC Systems Biology 3(1):54.

Lauritzen SL & Spiegelhalter DJ (1988) Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society. Series B (Methodological):157-224.

Lee SJ, et al. (2005) Metabolic engineering of Escherichia coli for enhanced production of succinic acid, based on genome comparison and in silico gene knockout simulation. Applied and environmental microbiology 71(12):7880.

Lenhard B, Hayes WS, & Wasserman WW (2001) GeneLynx: a gene-centric portal to the human genome. Genome Research 11(12):2151.

Levine A, Hu W, & Feng Z (2006) The P53 pathway: what questions remain to be explored? Cell Death & Differentiation 13(6):1027-1036.

Li Z & Chan C (2004) Inferring pathways and networks with a Bayesian framework. The FASEB Journal:304751.

Mihajlovic V & Petkovic M (2001) Dynamic bayesian networks: A state of the art. CTIT technical reports series.

Moloshok TD, et al. (2002) Application of Bayesian decomposition for analysing microarray data. Bioinformatics 18(4):566.

Moore AW (2001) Bayes Nets for representing and reasoning about uncertainty. Retrieved April 22:2008.

Mudunuri U, Che A, Yi M, & Stephens RM (2009) bioDBnet: the biological database network. Bioinformatics 25(4):555.

Murphy K (2001) The bayes net toolbox for matlab. Computing science and statistics 33(2):1024-1034.

Murphy K & Mian S (1999) Modelling gene expression data using dynamic Bayesian networks. University of California, Berkeley.

Murphy K & Weiss Y (2001) The factored frontier algorithm for approximate inference in DBNs.

Murphy KP (2002) Dynamic bayesian networks: representation, inference and learning. (Citeseer).

Murphy KP (2003) Dynamic bayesian networks. Probabilistic graphical models.

Myers C, et al. (2005) Discovery of biological networks from diverse functional genomic data. Genome biology 6(13):R114.

Myllymki P, Silander T, Tirri H, & Uronen P (B-Course: A web-based tool for Bayesian and causal data analysis.

Needham CJ, Bradford JR, Bulpitt AJ, & Westhead DR (2006) Inference in Bayesian networks. Nature biotechnology 24(1):51-54.

Needham CJ, Bradford JR, Bulpitt AJ, & Westhead DR (2007) A primer on learning in Bayesian networks for computational biology. PLoS Comput Biol 3(8):e129.

Neil M, Fenton N, & Nielson L (2000) Building large-scale Bayesian networks. The Knowledge Engineering Review 15(3):257-284.

Nikovski D (1998) Learning stationary temporal probabilistic networks. (Citeseer).

OCHS MF, MOLOSHOK TD, BIDAUT G, & TOBY G (2004) Bayesian decomposition: analyzing microarray data within a biological context. Annals of the New York Academy of Sciences 1020(The Applications of Bioinformatics in Cancer Detection):212-226.

Poole D (1993) Probabilistic Horn abduction and Bayesian networks. Artificial Intelligence 64(1):81-129.

Rodin AS & Boerwinkle E (2005) Mining genetic epidemiology data with Bayesian networks I: Bayesian networks and example application (plasma apoE levels). Bioinformatics.

Rogers S & Girolami M (2005) A Bayesian regression approach to the inference of regulatory networks from gene expression data. Bioinformatics 21(14):3131.

Rouillard JM, Zuker M, & Gulari E (2003) OligoArray 2.0: design of oligonucleotide probes for DNA microarrays using a thermodynamic approach. Nucleic acids research 31(12):3057.

Russell MJ & Bilmes JA (2003) Introduction to the special issue on new computational paradigms for acoustic modeling in speech recognition. Computer Speech & Language 17(2-3):107-112.

Sachs K, Gifford D, Jaakkola T, Sorger P, & Lauffenburger DA (2002) Bayesian network approach to cell signaling pathway modeling. Science's STKE 2002(148).

Schfer J & Strimmer K (2005) An empirical Bayes approach to inferring large-scale gene association networks. Bioinformatics 21(6):754.

Shachter RD (1998) Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). (Citeseer), pp 480–487.

Shafer G & Shenoy PP (Local computation in hypertrees. Work. Pap 201.

Silander T & Myllym‰ki P (2006) A simple approach for finding the globally optimal Bayesian network structure. (Citeseer).

Singh AP & Moore AW (2005) Finding optimal Bayesian networks by dynamic programming (Citeseer).

Singhal A & Brown C (1997) Dynamic Bayes net approach to multimodal sensor fusion. (Citeseer), pp 2-10.

Spiegelhalter DJ, Dawid AP, Lauritzen SL, & Cowell RG (1993) Bayesian analysis in expert systems. Statistical science 8(3):219-247.

Spirtes P, et al. (2001) Constructing Bayesian network models of gene expression networks from microarray data. (Citeseer).

Tanay A & Shamir R (2001) Computational expansion of genetic networks. BIOINFORMATICS-OXFORD- 17:270-278.

Troester MA, Hoadley KA, Parker JS, & Perou CM (2004) Prediction of toxicant-specific gene expression signatures after chemotherapeutic treatment of breast cell lines. Environmental health perspectives 112(16):1607.

Vigdis N, et al. (Validation of oligoarrays for quantitative exploration of the transcriptome. BMC Genomics 9.

Wang Y & Vassileva J (2003) Bayesian network-based trust model. (IEEE), pp 372-378.

Welch G & Bishop G (1995) An introduction to the Kalman filter. University of North Carolina at Chapel Hill, Chapel Hill, NC 7(1).

Wilkinson DJ (2007) Bayesian methods in bioinformatics and computational systems biology. Briefings in bioinformatics 8(2):109.

Woolf PJ, Prudhomme W, Daheron L, Daley GQ, & Lauffenburger DA (2005) Bayesian analysis of signaling networks governing embryonic stem cell fate decisions. Bioinformatics 21(6):741.

Yang JH, et al. (1998) Metacore: an application specific dsp development system. (ACM), pp 800-803.

Zhang Y, Deng Z, Jiang H, & Jia P (Dynamic Bayesian network (DBN) with structure expectation maximization (SEM) for modeling of gene network from time series gene expression data. BIOCOMP, eds., Hamid R. Arabnia and Homayoun Valafar:41ñ47.

Zou M & Conzen SD (2005) A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data. Bioinformatics 21(1):71.

Gokhul Krishna Kilaru

Blake St, Apt 720 C, Indianapolis, IN 46202
Phone (703) 944 – 8311, **Email: gokhulkrishnakilaru@gmail.com**

- Master of Science in Bioinformatics, Indiana University, Indianapolis, USA.
- Bachelor of Technology in Biotechnology, JNTU, Hyderabad, India.

CAREER:

Computational Biologist & Next-gen Sequencing Data Analyst Sep 2011-Present
National Institute of Allergy and Infectious Diseases
National Institutes of Health

EDUCATION:

Master of Science, Bioinformatics 2009-2011
IU School of Informatics, Indianapolis, IN **GPA- (3.6/4.0), Dean's List**
Thesis Advisor – Dr. Mathew J. Palakal

Bachelor of Technology, Biotechnology 2004-2008
Jawaharlal Nehru Technological University, India **GPA – (3.8/4.0)**

SKILL SET:

- **Languages & Scripts:** R, PERL, JAVA, SQL, PHP, HTML,CSS, XML, LAMP
- **Platforms:** UNIX, Win XP/2000/Vista/7, Macintosh OSX.
- **Tools:** FASTA, Blast, Clustal W, SPSS, Spotfire, BioMAP, Genome Browser, Ensembl, Waters Vision Publisher, Nugenesis SDMS, GENEGO, INGENUITY
- **Network Analysis Tools:** Pathway studio, Cytoscape, Visant, MetaCore, Gephi, Network workbench, Pajek, Cell Illustrator, Knowledge Editor
- **NGS Tools:** BWA, Samtools, GATK, SIFT and POLYPHEN
- **Database:** MySQL, Oracle, SQL Server 2005, MS Access and MS Excel

RESEARCH WORK EXPERIENCE:

- **Research Assistant** – Indiana University, School of Informatics (Jan 09 – Sep 11)
  - Thesis titled "Use of apriori knowledge in assessing the prediction performances of bayesian networks."
  - Gene expression and sequence data analysis obtained from GEO, GXD & SMD.
  - Mapping biological pathways to literature mined protein–protein associations.
  - Developed protein-protein interaction algorithm using literature mining protocols and machine learning methods.
  - Quality analyst, data mining specialist for protein-protein interaction algorithm.
  - Bioinformatics support and performance developer for Bio-Map tool.
  - Analysis of biological networks constructed from omics data.
  - Modeling biological pathways using different software tools.
  - Data analysis and visualization specialist for research articles.

- **Research Technician/Database Developer** – IU Dept. of Hepatology (Apr 09-Aug 09)
  - Database developer and interface designer for clinical research data.
  - Informatics support for smooth information retrieval, methodology enhancement and data flow.
  - Used MySQL, SQL Server, JAVA and MS ACCESS.

- **Data Analyst** – IU Dermatology, IU School of Medicine (Mar'09 – Feb'10)
    - Worked on collecting, analyzing and managing clinical data related to the on-going research in the department.

ACADEMIC ACTIVITIES:

- Presented a poster on "*Edge weighted networks and their marriage to biology*" at IUPUI
- Presented a poster on *"A web-system to identify novel protein-protein interactions"* at IUPUI.
- Presented a poster on "*Frontiers in Genetics and Biotechnology – Retrospect and Prospect"*, organized by Department of Genetics in 2007 at Osmania University, India.
- Presented a review paper on *"Molecular docking of colorectal cancer proteins"* at Osmania University, India.
- Presented review papers and posters at various National Level Biotechnology Symposiums.

PRACTICAL TRAINING

- Underwent in plant training in manufacturing facility at **Sipra Labs Limited**, India.

WORKSHOP

- Participated in a workshop held by **Indian Institute of Technology**, Chennai, Tamilnadu, India on "Nanotechnology".

ACTIVITIES

- Chairman (2011-2013), Media & PR, Telugu Association of North America (TANA), USA
- Media coordinator (2010 – Present), Greater Indianapolis Telugu Association (GITA)
- Chairman (2010-2011), Indian Students Advisory Council, IUPUI, Indianapolis, IN
- Co-chair (2009-2011), Media & PR, Telugu Association of North America (TANA), USA
- President (2009-2010), Indian Students Advisory Council, IUPUI, Indianapolis, IN
- Member of Heifer International, a volunteer organization at IUPUI.

LANGUAGES

- Fluent in English, Hindi, Telugu