ABSTRACT

BIG DATA ANALYTICS IN COMPUTATIONAL BIOLOGY AND BIOINFORMATICS

by
Kevin Byron

Big data analytics in computational biology and bioinformatics refers to an array of operations including biological pattern discovery, classification, prediction, inference, clustering as well as data mining in the cloud, among others. This dissertation addresses big data analytics by investigating two important operations, namely pattern discovery and network inference.

The dissertation starts by focusing on biological pattern discovery at a genomic scale. Research reveals that the secondary structure in non-coding RNA (ncRNA) is more conserved during evolution than its primary nucleotide sequence. Using a covariance model approach, the stems and loops of an ncRNA secondary structure are represented as a statistical image against which an entire genome can be efficiently scanned for matching patterns. The covariance model approach is then further extended, in combination with a structural clustering algorithm and a random forests classifier, to perform genome-wide search for similarities in ncRNA tertiary structures.

The dissertation then presents methods for gene network inference. Vast bodies of genomic data containing gene and protein expression patterns are now available for analysis. One challenge is to apply efficient methodologies to uncover more knowledge about the cellular functions. Very little is known concerning how genes regulate cellular activities. A gene regulatory network (GRN) can be represented by a directed graph in which each node is a gene and each edge or link is a regulatory effect that one gene has on another gene. By evaluating gene expression patterns, researchers perform *in silico* data analyses in systems biology, in particular GRN inference, where the “reverse engineering” is involved in predicting how a system works by looking at the system output alone.

Many algorithmic and statistical approaches have been developed to computationally reverse engineer biological systems. However, there are no known bioinformatics tools capable of performing perfect GRN inference. Here, extensive experiments are conducted to evaluate and compare recent bioinformatics tools for inferring GRNs from time-series gene expression data. Standard performance metrics for these tools based on both simulated and real data sets are generally low, suggesting that further efforts are needed to develop more reliable GRN inference tools. It is also observed that using multiple tools together can help identify true regulatory interactions between genes, a finding consistent with those reported in the literature. Finally, the dissertation discusses and presents a framework for parallelizing GRN inference methods using Apache Hadoop in a cloud environment.

BIG DATA ANALYTICS
IN COMPUTATIONAL BIOLOGY AND BIOINFORMATICS

by
Kevin Byron

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

May 2017

APPROVAL PAGE

BIG DATA ANALYTICS IN COMPUTATIONAL BIOLOGY AND BIOINFORMATICS

Kevin Byron

Dr. Jason T. L. Wang, Dissertation Advisor — Date
Professor of Computer Science, NJIT

Dr. James McHugh, Committee Member — Date
Professor of Computer Science, NJIT

Dr. David Nassimi, Committee Member — Date
Associate Professor of Computer Science, NJIT

Dr. Dimitrios Theodoratos, Committee Member — Date
Associate Professor of Computer Science, NJIT

Dr. Yi Chen, Committee Member — Date
Associate Professor of School of Management, NJIT

Dr. Katherine G. Herbert, Committee Member — Date
Associate Professor of Computer Science, Montclair State University

BIOGRAPHICAL SKETCH

Author:	Kevin Byron
Degree:	Doctor of Philosophy
Date:	May 2017

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2017
- Master of Science in Computer Science,
  Stevens Institute of Technology, Hoboken, NJ, 1987
- Bachelor of Science in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 1983
- Associate in Applied Science in Computer Science,
  Union County Technical Institute, Scotch Plains, NJ, 1975

Major: Computer Science

Presentations and Publications:

K. Byron, K. G. Herbert, and J. T. L. Wang. *Bioinformatics Database Systems*, Abingdon, UK: Taylor & Francis, 2017.

Y. Abduallah, T. Turki, K. Byron, Z. Du, M. Cervantes-Cervantes, and J. T. L. Wang, "MapReduce algorithms for inferring gene regulatory networks using an information-theoretic approach," *BioMed Research International*, 2017.

M. Vasavada, K. Byron, Y. Song, and J. T. L. Wang. "Genome-wide search for pseudoknotted non-coding RNA: a comparative study," In *Pattern Recognition in Computational Molecular Biology: Techniques and Approaches (Wiley Series in Bioinformatics)*, Hoboken, NJ: John Wiley & Sons, 2016.

K. Byron, J. T. L. Wang and D. Wen, "Genome-wide prediction of coaxial helical stacking using random forests and covariance models," *International Journal on Artificial Intelligence Tools*, Vol. 23, No. 3, 2014.

K. Byron, C. Laing, D. Wen and J. T. L. Wang, “A computational approach to finding RNA tertiary motifs in genomic sequences: a case study,” *Recent Patents on DNA & Gene Sequences*, Vol. 7, No. 2, pp. 115-122, 2013.

K. Byron, J. T. L. Wang and D. Wen, “Genome-wide search for coaxial helical stacking motifs,” *IEEE 12th International Conference on Bioinformatics & Bioengineering (BIBE)*, Cyprus, pp. 260-265, 2012.

K. Byron, M. Cervantes-Cervantes and J. T. L. Wang, “Biological informatics: data, tools and applications,” in *Computational Intelligence and Pattern Analysis in Biological Informatics*, (eds. U. Maulik, S. Bandyopadhyay and J. T. L. Wang), Chapter 3, Hoboken, NJ: John Wiley & Sons, Inc., pp. 59-69, 2010.

K. Byron, M. Cervantes-Cervantes, J. T. L. Wang, W. C. Lin and Y. Park, “Mining *roX1* RNA in drosophila genomes using covariance models,” *International Journal of Computational Bioscience*, Vol. 1, No. 1, pp. 22-32, 2010.

To Jutka.

ACKNOWLEDGMENT

I thank my mentors, my dissertation committee, NJIT, the CS department staff, the GSO staff, my friends, my family and the good Lord.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Gene Ontology as an Enabling Tool for Mining

Recently, there has been much discussion on how to report biology knowledge in information systems. While it is difficult to persuade various stakeholders to alter processes they have become used to, there is general consensus about the need for a common, concise vocabulary of biological terms. An ontology provides a vocabulary for representing and communicating knowledge about a topic [14]. The Gene Ontology (GO) Consortium was formed in 1998 to establish and maintain an ontology about gene information. Over time, additional biology and medical ontology groups formed and today they are collectively called the Open Biomedical Ontologies (OBO) group. The OBO Foundry consists of ontologies in various stages of maturity. Among the more mature ontologies are the Cell Ontology (CL), Gene Ontology (GO), Foundational Model of Anatomy (FMA) and the Zebrafish Anatomical Ontology (ZAO).

In 2000, the GO consortium was a joint project of three model organism databases: FlyBase, Mouse Genome Informatics (MGI) and the *Saccharomyces* Genome Database (SGD). The goal of the consortium was to produce a precisely defined, structured, common, controlled vocabulary describing the roles of genes and gene products in any organism. [8] Within the gene ontology (GO), the following ontologies developed:

- biological process ontology: a biological objective to which the gene or gene product contributes;
- molecular function ontology: the biochemical activity of a gene product;
- cellular component ontology: a place in the cell where a gene product is active.

The Gene Ontology (GO) (*http://www.geneontology.org* - last accessed on 4-3-2017) is a community bioinformatics resource [13]. Each GO entry has a unique numeric identifier. Table 1.1 lists twelve model organisms selected for targeted curation. Each organism is shown with the name of its respective database.

Gene ontology (GO) information exists as a publicly available flat file. The current version of the full GO data set (last accessed on 4-3-2017) is located at

ftp://ftp.geneontology.org/pub/go/ontology/go.obo.

Predetermined sets of GO terms, called GO Slims, are used to aggregate gene product information. GO slims may be created by users according to their needs, and may be specific to species or to particular areas of the gene ontology. Go Slims provided by the Gene Ontology Consortium (GCO) (last accessed on 4-3-2017) are described at

http://www.geneontology.org/GO.slims.shtml.

Table 1.2 identifies GO slims maintained by GOC curators and others. As a GO flat file evolves, the respective Go Slim is updated simultaneously. Users can create customized GO Slims using the OBO-Edit tool (last accessed on 4-3-2017) available at

http://oboedit.org/.

OBO-Edit is an open source ontology editor written in Java. As an example, there is a GO Slim data set for the yeast, *Saccharomyces cerevisiae*, genome. The current version of the GO Slim yeast data set (last accessed on 4-3-2017) can be downloaded from

http://www.geneontology.org.

In addition to OBO-Edit, another valuable GO utility is AmiGO. AmiGO is a web-based tool that provides access to all terms and annotations in the GO database [13]. AmiGO users can browse the ontology terms and search the annotations. AmiGO (last accessed on 4-3-2017) is available at

http://amigo.geneontology.org.

Each gene ontology entry is classified as a biological process, molecular function or cellular component. GO is used to compare computational biology experiments with wet lab results. GO is also useful in establishing gene regulatory networks (GRN). To map genes to GO terms, a mapping data set (last accessed on 4-3-2017) is provided by NCBI at

ftp://ftp.ncbi.nlm.nih.gov/gene/DATA/gene2go.gz.

The hypergeometric test can be used to identify whether or not a cluster of ncRNA genes has a better than average chance of possessing a specific GO trait. This is a valuable tool when analyzing unknown ncRNA genes. Khaladkar, *et al.*, [56] used the gene ontology to classify functions of ncRNA genes utilizing hypergeometric testing.

1.2 Biological Pattern Discovery

Pattern discovery is the problem of finding recurring patterns in biological data. Patterns can be sequential, mainly when discovered in DNA sequences. Patterns can also be structural (e.g., when discovering RNA motifs). Finding common structural patterns helps to understand cellular mechanisms, such as post-transcriptional regulation. Unlike sequentially conserved DNA motifs, RNA motifs exhibit conservation in structure, which may be common even if the sequences are different. Hundreds of algorithms have been developed to solve the sequential motif discovery problem, while less work has been done for the structural motif discovery case.

Finding recurring patterns, motifs, in biological data gives an indication of important functional or structural roles. Motifs can be either sequential or structural. Motifs are represented as sequences when they represent repeated patterns in biological sequences. Motifs are structural when they represent patterns of conserved

Table 1.1 Twelve Model Organisms Selected for Targeted Curation and Their Respective Databases

GO Species	**Species Database**
Arabidopsis thaliana	*Arabidopsis* Information Resource (TAIR)
Caenorhabditis elegans	WormBase
Danio rerio	Zebrafish Information Network (Zfin)
Dictyostelium discoideum	Dictybase
Drosophila melanogaster	FlyBase
Escherichia coli	EcoliHub
Gallus gallus	AgBase
Homo sapiens	Human UniProtKB-Gene Ontology
Mus musculus	Mouse Genome Information (MGI)
Rattus norvegicus	Rat Genome Database (RDG)
Saccharomyces cerevisiae	*Saccharomyces* Genome Database (SGD)
Schizosaccharomyces pombe	GeneDB *S. Pombe*

Table 1.2 List of GO Slims Maintained by GO Consortium as Part of the GO Flat File

GO Glim Name	**GO Slim Developer**
Generic GO slim	GO Consortium
Plant slim	The *Arabidopsis* Information Resource
Candida albicans	*Candida* Genome Database
Protein Information Resource slim	Darren Natale, PIR
Schizosaccharomyces pombe slim	Val Wood, PomBase
Yeast slim	*Saccharomyces* Genome Database
Aspergillus slim	*Aspergillus* Genome Data
Metagenomics slim	Jane Lomax and the InterPro group

base pairs (e.g., RNA secondary structures). Learning RNA structural motifs is needed to understand cellular mechanisms.

The structural motif discovery problem should not be confused with the two close problems: RNA structure prediction and RNA consensus structure prediction. In the former, it is required to predict the secondary structure of a single RNA sequence, whereas in the latter, it is required to find a list of base pairs that can simultaneously be formed in a set of related RNA sequences. Predicting structure involves minimizing total molecular free energy. Structures are evolutionary related and share a similar overall fold. Evolutionary conservation information is utilized to improve the accuracy of structure prediction process.

1.3 Biological Data Classification

Classification assigns data to predefined categorical class labels [45, 47]. A classification is an attribute or feature in a data set in which a researcher is most interested. It is defined as the dependent variable in statistics. To classify data, a classification algorithm creates a classification model consisting of classification rules. For example, automobile insurers have developed classification models to categorize drivers' applications as "risky" or "safe". In the medical field, classification can be used to help define medical diagnosis and prognosis based on symptoms, history of family illness and health conditions.

Classification is a two-step process consisting of training and testing. In the first step, a training model is built which consists of classifying rules. The training model is first "taught" the rules. Then, when the training model is "tested", i.e., presented with new data not from the training data, the training model is able to make a decision based on what it has learned from the training data.

A simplified hypothetical example of an RNA classification rule to determine "Topological Family" class label using features of an RNA loop structure is shown below:

```
IF LoopSize1 = LoopSize2 AND LoopSize3 ¡ Loopsize4
THEN
TopologicalFamily = "X"
ELSE
TopologicalFamily = "Y"
END-IF
```

Some classification rules use a mathematical formula to determine a class label. Classifying rules are not necessarily 100% true; generally, rules with 90-95% accuracy are regarded as solid rules. The accuracy of a classifier (or classification model) depends on the degree to which classifying rules are true.

The second step, testing, examines a classifier using testing data for accuracy. The class labels for the test data are known. The classifier is expected to predict the class label for each test case based on how it has been taught. Generally, the testing process is very simple and computationally inexpensive as compared to the training step, which may be complex and require considerable computational resources.

An interesting technique in classification is the ensemble approach. The rationale behind the ensemble approach is that multiple classifiers (or classification models) working together can yield better classification accuracy than the use of a single classifier. As a simple example, if Classifiers A, B, and C predict that a hard-to-classify patient (patient1) has a disease and Classifiers D and E predict that patient1 doesn't have that disease, then, by using a voting strategy, the ensemble predictor would predict that patient1 has the disease. In some cases, each classifier may be assigned different weights and the final ensemble prediction would then be a weighted average of the classifier votes. The classification case study in this chapter explores the ensemble method in greater detail.

1.4 Biological Data Clustering

Clustering is defined as unsupervised learning that occurs by observing only independent variables (unlike supervised learning analyzing both independent variables and dependent variables) [45, 47, 107]. In order words, unlike classification, clustering does not use "class". In fact, this is the main difference between classification and clustering. For this reason, clustering may be best used for studies of an exploratory nature, especially if those studies encompass a large amount of data, but very little is known about data (such as the mass of data typically generated by microarray analysis).

Clustering is used to group objects into a specific number of clusters so that the objects within a cluster have very high similarity and objects from different clusters have very low similarity. Similarities between two objects are measured using their attribute values. A very early application of clustering in biology was to cluster similar plants and animals to create taxonomies based on their attributes (such as the number of petals and the number of legs). A number of clustering algorithms have been introduced and used over the last few decades. These algorithms are mainly categorized into hierarchical and partitional. Each category of clustering methods will be further discussed in the following sections.

Hierarchical agglomerative clustering algorithms successively merge the most similar two groups of objects based on the pairwise distances between two groups of objects until a termination condition holds, so that objects are hierarchically grouped. For this reason, hierarchical algorithms themselves can be effectively categorized according to the respective methods of calculating the similarity (or distance) between two groups of objects. In order words, this categorization is based on how the representative object of each group for similarity calculation is selected.

While hierarchical clustering is agglomerative, i.e., starting with atomic elements and aggregating them into clusters of increasing size, divisive clustering starts with the a complete data set and sub-divides the data set into smaller partitions. A divisive clustering algorithm iteratively performs these two sub-problems: 1) decide the best cluster to be split; 2) decide the best way to split the selected cluster [95].

Unlike hierarchical clustering algorithms, partitional clustering algorithms require a user to input a parameter k, which is the number of clusters. Generally, partitional algorithms directly relocate objects to k clusters. Partitional algorithms are categorized according to how they relocate objects, how they select a cluster centroid (or representative) among objects within a (incomplete) cluster, and how they measure similarities between objects and cluster centroids. For example, k-means, the most widely-used partitional algorithm, first randomly selects k centroids (objects), and then decomposes objects into k disjoint groups by iteratively relocating objects based on the similarity between the centroids and the objects. In k-means, a cluster centroid is the mean value of objects in the cluster. In many cases, the cluster centroids are not actual cluster objects. Unlike k-means, k-medoids selects the nearest object to the mean value of objects in a cluster.

The major advantage of partitional clustering algorithms over other methods, is their superior clustering accuracy as compared with hierarchal clustering algorithms that is the result of their global optimization strategy (i.e., the recursive relocations of objects). In addition, partitional algorithms can handle large data sets which hierarchal algorithms cannot (i.e., better scalability) and can more quickly cluster data. In short, partitional algorithms are more effective and efficient than hierarchical algorithms. One major drawback to the use of partitional algorithms is that their clustering results depend on the initial cluster centroids to some degree because the centroids are randomly selected. Thus, clustering results obtained are a little different each time the partitional algorithm runs. Such a process is known as non-deterministic. A deterministic process, by comparison, will yield the same result each and every time it is run.

1.5 Biological Network Inference

A simplification of the "central dogma of biology" is this: DNA begets RNA; RNA begets protein. When a gene in your DNA is "expressed" for any number of reasons, the result is an RNA molecule called a transcript, and that process is called transcription. Some of those RNA transcripts are converted into protein. Researchers believe that most of the RNA transcripts are not converted to protein and that they serve a variety of other functions in the cell. RNA research is currently very active. The collective sum of all genes of an organism is known as its genome, and the study of the genome is called genomics. The collective sum of all expressed RNA transcripts in the cell at a point in time for an entire genome is known as a transcriptome. The study of the transcriptome, then, is called transcriptomics. Massive databases have been established to gather transcriptomes for research into the reasons for gene expressions. There is evidence that some genes cause other genes to be expressed. This "cause-and-effect" relationship can be described in the form of a Gene Regulation Network (GRN), which in many cases can be represented by a simple graph. Much research is currently focused on the study on the transcriptome at regular time intervals in an attempt to "infer" a reliable GRN.

1.6 Biological Data Mining in the Cloud

Advances in computational biological analytics tools must keep pace with biological Big Data, i.e., the exponential growth of biological genomic data. Cloud computing is “a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.” (*http://csrc.nist.gov/* - last accessed on 4-3-2017).

Cloud-based tools with massive parallel-processing capability can be used for modern computational biological research on Big Data. Cloud computing model offers an alternative to the expensive computational and storage needs of research computations [43]. Pair-wise DNA or RNA sequence alignment, distance calculation, clustering, multidimensional scaling and visualization of gene sequences are computationally intensive operations. Similarly computationally expensive are interactive parallel computations that can be used to determine differential equation parameters in resolving gene network inference problems.

A Big Data problem can be solved by breaking it into smaller parts, solving the smaller parts simultaneously in parallel, and then gathering together the individual results. The time saved with this ”parallel processing” approach is called Speed-Up. Amdahl’s law (Eq. 1.1) defining “Speed Up” is often used in parallel computing to predict the theoretical maximum speedup using multiple processors [7].

$$Amdahl'sLaw : SpeedUp = \frac{1}{(1-P)+\frac{P}{S}} \tag{1.1}$$

In Eq. 1.1, P is the proportion of a problem that can be divided into parallel computing tasks, and $1 - P$ is the proportion that cannot be parallelized, i.e., the “sequential” proportion. The maximum acceleration capability lies in S, meaning that P can be accelerated S times by using S parallel processors. The speedup of a program using multiple processors in parallel computing is limited by the time needed for the sequential fraction of the program. For example, assume that a program needs 20 hours using a single processor core. Assume that a particular portion of the program (i.e., $1 - P$) which takes one hour (i.e., 5%) to execute cannot be parallelized. The remaining 19 hours (i.e., 95%) of execution time can be parallelized (i.e., P). In this case, regardless of how many processors are devoted to a parallelized execution of this program, the minimum execution time cannot be less than one hour. Hence the speedup is limited to at most 20. Amdahl's law is concerned with the speedup achievable by "parallelizing" proportion P of a computation where the improvement has a speedup of S. Amdahl's Law clearly describes the efficiency and limitation of parallel computing and has been widely adopted.

MapReduce [27] is a distributed data processing software development framework developed by Google to support parallel distributed execution of its data intensive applications. The MapReduce approach uses "divide-and-conquer" to speed up the processing of Big Data. Using MapReduce, a data processing solution consists of map and reduce steps. Google uses this framework internally to execute thousands of MapReduce applications per day, processing peta-bytes of data, all on commodity hardware. Running between the map and reduce phases is an internal shuffle phase for handling intermediate results. The MapReduce framework automatically executes those functions in parallel over any number of processors.

The MapReduce framework contains two main phases, map and reduce, that are controlled by a driver program in one designated "master" machine. In the map phase, the driver forwards portions of input data to multiple computing nodes designated as "slave" machines. The MapReduce driver instructs the computing nodes to perform calculations according to a user-defined program for the map process (AKA mapper). The map results are saved to immediate files.

In the reduce phase (after the calculations on mappers have been completed), the driver directs the computing nodes to gather the results from the intermediate files. These intermediate results are processed using a user-defined program for the reduce step (AKA reducer). Then reducers contribute their results to form the complete output.

MapReduce provides an easy-to-use programming model that features fault tolerance, automatic parallelization, scalability and data locality-based optimizations. Many scientific computation algorithms that rely on iterative computations can be implemented with a MapReduce computation specified for each iterative step.

One major factor in the success of MapReduce is the innovative distributed file system developed by Google called Google File System (GFS) [37]. GFS is highly fault tolerant to due extensive data replication on inexpensive commodity storage equipment.

The MapReduce programming model provides an easy-to-implement framework with fault tolerance capabilities. This model has been used to successfully solve many large-scale scientific computing problems, including problems in the life sciences. The goal of MapReduce is to deploy a large amount of time- and memory-consuming tasks to many computing nodes that process tasks in parallel running user-defined algorithms. The flow of the MapReduce process involves one master machine and many slave machines running MapReduce tasks as directed by the driver process in the master machine.

Apache Hadoop (*http://hadoop.apache.org* - last accessed on 4-3-2017) is a widely used open-source implementation of the Google MapReduce [27] distributed data processing framework. Apache Hadoop uses the Hadoop Distributed File System (HDFS) for data storage, which stores the data across the local disks of the computing nodes while presenting a single file system view through the HDFS application program interface (API). HDFS is an open-source implementation of GFS. Like GFS, HDFS is intended to be run on commodity storage equipment. HDFS provides the distributed file system support to help ensure high performance and fault tolerance. High levels of reliability are achieved through data replication. Throughput

Table 1.3 List of Current Cloud-Based Platforms Used for Bioinformatics Research and Their URLs (last accessed on 4-3-2017).

Platform	**URL**
Amazon Web Services (AWS)	*http://aws.amazon.com/*
Eoulsan	*http://transcriptome.ens.fr/eoulsan/*
NIH Biowulf	*http://hpc.nih.gov/systems/*
The Galaxy Project	*http://galaxyproject.org/*

performance is optimized by scheduling data transfer from the data replica nearest to the location of the computation node. Apache Hadoop performs duplicate executions of slower tasks and handles failures by rerunning the failed tasks using different worker machines.

Cloud-based biological research is currently very active. Stormbow, mentioned previously, analyzes RNA nucleotide sequence (i.e., RNA-Seq) samples using cloud-based resources [129]. AWS was the infrastructure used to develop Stormbow. Stormbow took 6 to 8 hours to process one RNA-Seq sample that contained over 100 million individual RNA molecules. The average cost was $3.50 per sample.

Bioinformatics is being challenged by increasingly larger data sets leading to computational jobs that take unacceptably long times if done on a small number of machines. For these cases, distributed computing on multiple clusters at different locations is becoming an attractive, if not necessary, approach to achieve short run times. Table 1.3 lists cloud-based platforms available for biological data processing. Galaxy [38] for instance, is a powerful open-system web-based platform for data intensive biomedical research. Galaxy allows the user to perform, reproduce, and share complete analyses on the freely available server or on a private server instance.

CHAPTER 2

BIOLOGICAL PATTERN DISCOVERY: CASE STUDIES

2.1 Introduction

Two genome-wide biological data mining case studies are presented below. The first case study is an example of biological pattern discovery where an RNA secondary structures are evaluated using a covariance model. The second case study is an example of biological data classification where RNA tertiary structures are evaluated using a random forests classifier.

2.2 A Case Study in RNA Secondary Structure Data Mining

As an example of biological pattern discovery, a non-coding RNA pattern discovery case study is presented [20]. Evolutionarily conserved functional domains of non-coding RNA on chromosome X (*roX1*) of the fruit fly have been identified in eight *Drosophila* species. Interestingly, within the *roX1* RNAs of these same *Drosophila* species, conserved primary sequences were also found. Specifically, three repeats of the nucleotide sequence GUUNUACG were localized in the 3'-end of the predicted *roX1* RNAs for these eight *Drosophila* species. In this case study, a covariance model (CM) was used to search for the characteristic features of *roX1* functional domains as a way to classify new examples of these structured RNAs in other *Drosophila* species. In spite of high levels of genomic sequencing activities worldwide, annotation of the *Drosophila* species is still incomplete. Much chromosome coordinate information remains unknown. Therefore, whole genomes of *Drosophila* were obtained and scanned to identify results in available annotated regions, i.e., chromosomes or scaffolds. Using known *roX1* examples for comparative support, it is believed to be possible to predict novel *roX1* functional domains accurately from sequence information alone. Annotating *roX1* on a genomic scale provides insight into evolutionary processes among various species. The results of this case study indicate that a CM search and classification process is effective in mining *roX1* RNA genes. Furthermore, due to the flexibility of the CM search methodology, this mining approach may very likely prove successful for similar searches in various other organisms.

2.2.1 Introduction

Non-coding RNAs (ncRNAs) are functional RNA transcripts that are not translated into protein (i.e., they are not messenger RNAs). Research has shown that ncRNAs perform a wide range of functions in the cell [25, 29, 77, 102]. RNA on the X chromosome (*roX1*) plays an essential role in equalizing the level of transcription on the X chromosome in *Drosophila* males and females. Like humans, the *Drosophila* male has a single X chromosome while the *Drosophila* female has two X chromosomes [86]. Experiments have confirmed that *roX1* RNA exists in eight *Drosophila* species [85, 84, 83]. It is believed that there may exist secondary structure conservation of the *roX1* gene among other *Drosophila* species [84, 83]. Advances in the research of genomes from twelve *Drosophila* species [101] might help shed light on this interesting issue.

A highly regarded covariance model (CM) method named Infernal has been successfully used in the classification of ncRNAs. Infernal is considered by many bioinformaticians to be one of the most accurate tools for this purpose [99, 113]. Infernal is a genome-wide search tool, which applies stochastic context-free grammars expressed as CM's to find genomic regions that may contain ncRNAs [20, 105, 116]. A CM is a statistical representation of a group of RNAs that share a common consensus secondary structure [28]. The Infernal software package [41, 42, 82] contains a number of powerful utilities. One of these utilities, named CMbuild, creates a CM from a Stockholm alignment of sequences. Another Infernal utility, named CMsearch, is used to scan a genome for sequences that match to the model and classify candidates as likely or unlikely to belong to the group that the CM represents. The time to run a CMsearch process can be lengthy depending on the size of the genome scanned. However, by utilizing parallel processing methods, results can be obtained in considerably shorter timeframes.

The first step in this case study was to demonstrate the capability of using a CM in a genome scale homology search. Known *Drosophila roX1* sequences from eight species were gathered. These eight species of *Drosophila* are named *D. ananassae*, *D. erecta*, *D. melanogaster*, *D. mojavensis*, *D. pseudoobscura*, *D. simulans*, *D. virilis* and *D. yakuba* [84]. Using a "leave-one-out" testing approach, covariance models were created using seven species at a time and the genome of the eighth species was scanned for a match. This demonstrated that the CM classification process is feasible in that 6 of the 8 searches were successful. The next step was to scan the genomes of *Drosophila* species for which there are no confirmed *roX1* sequences. These four *Drosophila* species are named *D. grimshawi*, *D. persimilis*, *D. sechellia*, and *D. willistoni*. Such a comparative genomics approach has been successful in the unicellular organism Saccharomyces cerevisiae [55], i.e., a species of yeast very commonly used in winemaking, baking and brewing.

Using this CM approach, the results show strong evidence of the presence of *roX1* functional domains in the genome of *D. sechellia*. This finding is believed to be novel and significant in ongoing genomic studies of *Drosophila* and related taxonomic groups. This bioinformatics study lays the groundwork for future CM ncRNA classification.

2.2.2 Methods

Eight *roX1* RNA sequences were obtained use "wet lab" experiments, i.e., confirmed vs. predicted "in silico". These sequences were from eight *Drosophila* species named *D. ananassae*, *D. erecta*, *D. melanogaster*, *D. mojavensis*, *D. pseudoobscura*, *D. simulans*, *D. virilis* and *D. yakuba.* [84] Table 2.1 illustrates selected sequences from this group. In all eight cases, the sequences were expressed in standard FASTA format. The *roX1* RNA sequences are fairly large ranging is length from 3,433 to 3,768 nucleotides. The classification analysis process in this case study dictated that the large sequences be subdivided into smaller subsequences while preserving all species and position information for each subsequence. The RSmatch software package used in this case study has this sequence subdividing capability [70].

The eight *roX1* sequences were assigned names or tracking purposes as follows: in columns 1, 2 and 3, "yp1"; and in columns 4 and 5, a sequential 2-digit number. Starting and ending positions for each of the eight sequences were described in FASTA notation as "start:end" where "start" represents the first numeric position and "end" represents the last numeric position. For each of the original sequences, "start" had the value of 1 and "end" had the value of the length of the sequence. The starting and ending positions for each sequence were formatted specifically for RSmatch [70]. Then, when subsequences were extracted, the original FASTA notation was preserved and additional subsequence position information was inserted for sequence accurate tracking purposes.

To illustrate with an example, one FASTA sequence, named yp101, input to the RSmatch slide-and-fold process was annotated this way: ">yp101 (1:3493) droana *rox1*". Note that the length of this *roX1* gene sequence is 3,493 nucleotides (nt). RSmatch was used to extract 100 nt subsequences with 50 nt overlaps from this yp101 sequence. RSmatch produced properly annotated FASTA format sequences such as ">yp101:1-100 (1:3493) droana *rox1*", ">yp101:51-150 (1:3493) droana *rox1*", etc. Note that this notation clearly represents a 100 nt sequence extracted from positions 1 through 100 and positions 51 through 150 of the original yp101 sequence. All of the original FASTA notation information was retained in the FASTA notation of each subsequence. Providing position information in the notation of the

Table 2.1 Selected *Drosophila roX1* Sequences (Source: www.flybase.org; Cr = chromosome; Sc = scaffold)

Species	Length	Region	Coordinates
D. yakuba	3,433	Cr X	4658396 - 4661828
			3710814 - 3710795
D. simulans	3,439	Cr X	2761962 - 2759151
			2762379 - 2761996
			2759122 - 2758943
			9903425 - 9903446
D. erecta	3,462	Sc 4690	1139892 - 1137083
			1140318 - 1139928
			1137036 - 1136857
D. melanogaster	3,468	Cr X	3755987 - 3754338
			3754043 - 3753143
			3756379 - 3756024
			3754304 - 3754082
			3753108 - 3752929

extracted subsequences is a critical function performed by RSmatch. In a similar manner, all eight *Drosophila roX1* sequences evaluated for this work were annotated for compatibility with RSmatch, thus preserving subsequence position information. RSmatch slide-and-fold process was run with the following parameters: sequence size = 100 nt; overlap size = 50 nt; minimum free energy = 0. RNA structures were prepared using the RSmatch "slide and fold" method. For each sequence, 100 nt subsequences were extracted at every 50 nt position from the 5'-end downstream to the 3'-end resulting in consecutive subsequences overlapping with one another on a 50 nt segment. Subsequences shorter than 100 nt, i.e., at the 3'-end, were also kept. All subsequences were then "folded" using the RNAsubopt function in the Vienna RNA package [50] with the setting "-e 0". The Vienna RNA package is highly regarded for its ability to accurately predict the secondary structure of non-coding RNA using the minimal thermodynamic energy approach. With this "-e 0" setting, multiple folding structures that have the same minimum thermodynamic energy are generated. Using this method with RSmatch, 773 structures were obtained from the eight original *Drosophila roX1* sequences.

RSmatch was used to conduct pairwise comparisons of all 773 RNA structures produced in the process previously described. In this step, RSmatch was configured for nucleotide matching scores of 1 and 3 in single-stranded (ss) and double-stranded (ds) regions, respectively. In addition, mismatch scores configuration settings were -1 and 1, in ss and ds regions, respectively. The gap penalty was -6 for both ss and ds regions. This scoring scheme essentially gave more weight on matches in ds regions than those in ss regions. Three unduplicated FASTA sequences were identified and extracted from the highest-scoring pairwise alignments.

The MXSCARNA [106] package was used to align sequences for the covariance model (CM) used in the case study. The resulting alignment was rendered in the

Stockholm format with predicted structure annotation. This alignment was input to the Infernal package utility, named CMbuild, to create a CM.

The CM search utility, CMsearch, was run against a dataset of *Drosophila* FASTA sequences. The genomes from 12 *Drosophila* species (i.e., *D. ananassae*, *D. erecta*, *D. grimshawi*, *D. melanogaster*, *D. mojavensis*, *D. persimilis*, *D. pseudoobscura*, *D. sechellia*, *D. simulans*, *D. virilis*, *D. willistoni* and *D. yakuba*) were downloaded from Indiana University's FlyBase database [23] (Table 2.2). Many *Drosophila* genomes have not yet been completely annotated into clearly defined chromosomes. As part of active research and sequencing efforts, the annotation of *Drosophila* and other genomes become richer and more informative.

The Infernal package (version 1.0) utility, CMsearch, was used to locate structures in *Drosophila* genomes with probability of matching the constructed CM [82]. To improve computational efficiency, large FASTA sequences were subdivided into smaller, overlapping subsequences to facilitate independent parallel searching without negatively impacting results. Using a stochastic dynamic programming algorithm, Infernal located and reported secondary structures in *Drosophila* genomes similar to the profile that the CM represents. Given the structural similarity and high score result of the CM search, a *D. sechellia* sequence discovered in the genome scan was predicted to represent *roX1* functional domain characteristics.

2.2.3 Results

The objective of this case study was to classify functional structure elements, i.e., non-coding RNA, in genomes of *Drosophila* species as potential *roX1* homologues. To an extent, portions of an approach previously deployed [57, 56, 58, 70] were used as models in this case study. First, eight sequences of *roX1* RNA transcripts were obtained (Table 2.1). Next, a "slide and fold" method to construct RNA structures was executed, as described in Methods. In this approach, subsequences 100 nucleotides (nt) in length or shorter were folded according to their thermodynamic properties using the Vienna RNA package [50]. Adjacent subsequences were overlapped by 50 nt. Non-coding RNA structures can be predicted accurately and efficiently in this way for two reasons:

- prediction for small ribonucleotide structures is more accurate and efficient than for large ones;
- structures with a size smaller than 50 nt were folded twice as subsequences of two different larger structures, further increasing the probability of obtaining accurate RNA structure predictions.

The Vienna folding package was run with a configuration that yielded multiple RNA structure predictions with the same minimum free energy for a given sequence to further improve folding accuracy. This step resulted in 773 predicted RNA structures.

Species vs. species pair-wise comparisons were applied using all 773 predicted RNA structures. For computational efficiency, each alignment was run on a separate processor independent of all others using a high performance computing (HPC) cluster, leveraging parallel processing speed-up capabilities [88]. This HPC system, a Sun Microsystems Discovery cluster, has 112 AMD Opteron dual-core Linux nodes with 2 GB of RAM per node. The operating system used was Red Hat Enterprise Linux AS release 4 Update 8. In this manner, approximately 520,000 pair-wise alignments were completed in less than five minutes. Each comparison yielded an alignment score. A group of three structures that were scored similarly and had similar lengths were selected. At this step RNA structures were obtained from *D. melanogaster*, *D. simulans* and *D. yakuba.*

A covariance model (CM) was created from this group of three structures by first aligning the sequences into the Stockholm format and then running the Infernal CMbuild utility. The complete genomes of eight *Drosophila* species for which the presence of *roX1* ncRNA transcripts has been confirmed were used as targets in CM searches. All complete genomes used in this study were obtained from Indiana University's FlyBase database (*http://www.flybase.org* - last accessed 0n 4-3-2017)

Table 2.2 Description of Twelve *Drosophila* Genomes Downloaded from FlyBase Public Database

Species	**Release**	**Date**	**Nucleotides**	**Files**
D. melanogaster	5.18	5/16/2009	130,430,583	7
D. simulans	1.3	7/24/2008	137,828,247	1
D. erecta	1.3	7/24/2008	152,712,140	1
D. pseudoobscura	2.4	5/19/2009	152,738,921	1
D. yakuba	1.3	7/24/2008	165,693,946	1
D. sechellia	1.3	7/24/2008	166,577,145	1
D. persimilis	1.3	7/24/2008	188,374,079	1
D. mojavensis	1.3	7/24/2008	193,826,310	1
D. grimshawi	1.3	7/24/2008	200,467,819	1
D. virilis	1.2	7/24/2008	206,026,697	1
D. ananassae	1.3	7/24/2008	230,993,012	1
D. willistoni	1.3	7/24/2008	235,516,348	1

[23]. These genomes were the most current releases at the time the study was conducted (Table 2.2). A CM search located the *roX1* genes precisely where they were known to be present in six *Drosophila* species, i.e., *D. ananassae*, *D. erecta*, *D. melanogaster*, *D. pseudoobscura*, *D. simulans*, and *D. yakuba*. However, the CM search failed to locate the known *roX1* ncRNAs on the remaining two *Drosophila* species, i.e., *D. mojavensis* and *D. virilis*. In five of the six successful searches, the highest scoring search result represented a sequence within the known range of the *roX1* genomic coordinates for that species. The sixth successful search, on *D. pseudoobscura*, produced the third highest scoring search result that represented a sequence within the known range of the *roX1* genomic coordinates for that species. The two highest scores for *D. pseudoobscura* likely represent sequences with conserved *roX1* functionality.

For computationally efficiency, the downloaded genome files were separated into smaller files of 2 million base pairs (Mbp) per file. FASTA sequences larger than 2 Mbp were split into smaller FASTA sequences which overlapped one another by 5 thousand base pairs (Kbp) to prevent loss of accuracy in the study. This approach is similar to the RSmatch slide-and-fold approach described in Methods. However, this process was performed with custom Perl scripts. Concurrent CMsearch jobs were run against multiple genome sequences in parallel using an high performance computing (HPC) cluster. In this manner, a covariance model (CM) search of an entire genome took about 10 minutes.

This case study then focused on classifying potential *roX1* functional structure elements in the genomes of the four fully sequenced *Drosophila* species in which the presence of *roX1* transcripts had not yet been confirmed. These four species are named *D. grimshawi*, *D. persimilis*, *D. sechellia* and *D. willistoni*. The most current release of these complete genomes were obtained from the FlyBase database. [23]. The same CM previously used was used to search for presence of *roX1* functional domains. While scoring results were not significant for three of the four species, a strong score resulted from the CM search on the *D. sechellia* genome (Table 2.3). This high score shows strong evidence of a *roX1* functional domain in a specific area of the *D. sechellia* genome, namely scaffold_4. Furthermore, in spite of the *D. sechellia*'s incomplete annotation, this result likely indicates that this region of the genome may be located in the X chromosome of *D. sechellia*. These findings need to be confirmed by wet lab experiments.

To investigate possible *roX1* homology between species, *roX1* gene sequences FBgn0019661 (for *D. melanogaster*) and FBgn0255860 (for *D. sechellia*) were downloaded from the FlyBase database. A pair-wise alignment on the two sequences was performed. Using the program DiAlign [104] with the "-n" option for nucleic acid sequence comparison, a result of 94% similarity between the two gene sequences was shown. This result indicates high probability of conserved *roX1* functionality between the two species.

In this case study, a systematic and computationally efficient approach was designed and developed to classify *roX1* RNA structure elements conserved in *Drosophila* species. This approach consists of three major steps:

- comparison of RNA structures among all *roX1* RNAs;
- selection of RNA structure groups significantly associated with those in other species;
- utilization of a highly regarded structure-searching methodology (i.e., covariance models) which, in addition to being highly sensitive and specific, is also very flexible.

The stochastic representation of a cluster of RNA structures can be fine-tuned as needed by adding or removing structures from the cluster. Using parallel processing contributes to overcoming the burden of lengthy processing times. This method was applied to classifying small RNA structures chiefly because these structure can be classified more accurately compared with the methods that only use thermodynamic minimization. As more powerful RNA structure classification and prediction programs become available, this case study approach can be extended to larger RNA structures.

To compare the effectiveness of different search tools, BLAST search was compared with Infernal in a search for conserved structural motifs. Since BLAST is not designed to detect covariant base pairs that are critical in an RNA secondary structure, Infernal was expected to perform better than BLAST. Each of the three sequences from the case study's covariance model were used in FlyBase BLAST to search for homologues in the complete genomes of all 12 *Drosophila* species downloaded from FlyBase. Every homologue detected by BLAST was also detected by Infernal. However, BLAST failed to detect *roX1* evidence in *D. ananassae* and *D. pseudoobscura*, while such evidence was detected by Infernal. This simple experiment provides an insight into the complexity involved in the classification of ncRNA motifs.

Table 2.3 Summary of Homologues Found in Seven *Drosophila* Species Showing CM Score, FlyBase Region, Region Coordinates, Strand and *roX1* Status for Each Homologue

Genome	**Score**	**Region**	**Coordinates**	**Strand**	***roX1*?**
D. melanogaster	88.84	chromosome X	3753295 - 3753232	-	Y
D. yakuba	88.69	chromosome X	4661475 - 4661538	+	Y
D. simulans	88.1	chromosome X	2759303 - 2759240	-	Y
D. sechellia	88.1	scaffold 4	2954091 - 2954154	+	N/A
D. erecta	72.78	scaffold 4690	1137235 - 1137172	-	Y
D. ananassae	32.26	scaffold 13117	692432 - 692373	-	Y
D. pseudoobscura	29.4	Unknown group 410	14965 - 14898	-	N
D. pseudoobscura	28.28	XL group1e	6898105 - 6898042	-	Y
D. pseudoobscura	29.11	Unknown group 260	63165 - 63089	-	N

By conducting an homology scan on a complete genome or species chromosome, a researcher can confirm whether a functional domain is present throughout that genome or species chromosome, respectively. A stem-loop structure was previously predicted in *roX1* RNA on the X chromosome of *D. melanogaster* [103], for instance, and it was determined that this structure was conserved in several species of *Drosophila* [84]. This case study confirms that among seven different *Drosophila* species, the *roX1* functional domain is only present on the X chromosome and is also absent from all chromosomes other than X. The maturation of genome annotation and the translations of scaffold regions into chromosome regions will whether this observation continues to hold.

2.2.4 Conclusion

In this case study, RSmatch and Infernal were demonstrated to be effective tools in discovering patterns of novel ncRNAs. Homology searching is common in bioinformatics, yet some of the most popular homology search methods such as BLAST and FASTA, are often the least accurate [32]. For non-coding RNA, homology searching is more challenging compared with a sequence homology search. This is due to intramolecular covariant base pairs in ncRNA that are conserved to a higher degree with respect to their primary structure, i.e., their nucleotide sequence.

An Infernal search requires considerable computer run time [82]. Freyhult, *et al.*, estimated that with a search query for the transfer RNA (tRNA) type of ncRNA, Infernal would take about 96 days to search the entire human genome on a single processor [32]. Innovative methodologies including HMM filtering and sequence-based heuristics [114, 125] have been employed to improve computational efficiency. In this study, as described, parallel processing with a high performance computing cluster was used for speed-up and improved throughput.

Whole genomes of all 12 sequenced species of *Drosophila* were scanned. All 12 species are believed to have a common ancestor that existed about 40 million years ago [84]. Phylogenetic relationships are based on the premise that species that evolved "relatively" recently will have more genetic similarities than those species that evolved earlier. As a result of this case study, the presence of the *roX1* ncRNA is verified as previously reported by other authors in six *Drosophila* species. In addition, strong evidence of the presence of *roX1* in *D. sechellia* was found, which was not known to be previously reported.

2.3 A Case Study in RNA Tertiary Structure Data Mining

As an example of biological data mining, a case study in the mining of a three dimensional RNA motif is presented. Artificial intelligence tools are used to find motifs in DNA, RNA and proteins. In this case study, a computational tool for finding RNA tertiary motifs in genomic sequences was designed and developed. Specifically, this tool predicted genomic coordinate locations for coaxial helical stackings in 3-way RNA junctions. These predictions were provided by CSminer, a tertiary motif search package that utilized two versatile methodologies: random forests and covariance models. A coaxial helical stacking tertiary motif occurs in a 3-way RNA junction where two separate helical elements are aligned on a common axis to form a pseudocontiguous helix which provides thermodynamic stability to the RNA molecule. The CSminer tool used a genome-wide search method based on covariance models to find a genomic region that may contain a coaxial helical stacking tertiary motif. CSminer also used a random forests classifier to predict whether the genomic region indeed contains the tertiary motif. Experimental results demonstrated the effectiveness of the CSminer approach.

2.3.1 Introduction

It is important for bioinformaticians to develop pattern discovery tools that leverage increasingly powerful computational methodologies in critical life science research. In this case study, CSminer (i.e., Coaxial helical Stacking miner), predicted locations, i.e., genomic coordinates, of coaxial helical stackings in genomes. A coaxial helical stacking occurs in an RNA tertiary structure where two separate helical elements are aligned on a common axis and form a pseudocontiguous helix [62] at an RNA junction. An RNA junction is an important non-coding RNA (ncRNA) loop structure that forms where three or more helices meet. Coaxial helical stacking tertiary motifs may occur in several large RNA structures, including group II introns [109], large ribosomal subunits [9, 96, 115], pseudoknots [1], and transfer RNA (tRNA) [60], Coaxial helical stackings provide thermodynamic stability to the RNA molecule [59, 112], and reduce the separation between loop regions within junctions [2]. Coaxial helical stacking interactions are also involved in long-range interactions in many RNAs [119] and are essential features in a variety of other RNA junction topologies.

In this case study, the focus was on the 3-way RNA junction, though many RNA junctions exist in 4-way and higher forms. The topologies of known 3-way RNA junctions have been studied extensively [67]. 3-way RNA junctions that contain a coaxial stacking are classified into three topological families called A, B and C, depending primarily on the orientation of the helix that is not involved in the coaxial stacking and on the lengths of the unpaired base regions separating the helices. Figure 2.1 illustrates these three topological families.

Each 3-way RNA junction in Figure 2.1, has three helices labeled P1, P2 and P3. In a helix region, bases are paired in standard Watson-Crick pairings. In all three examples in the figure, P1 and P2 are presumed to be coaxially stacked. There is no presumption in any of these examples about the positions of the 5'-end and the 3'-end of the RNA molecule. In bioinformatics literature, the helix in a 3-way RNA

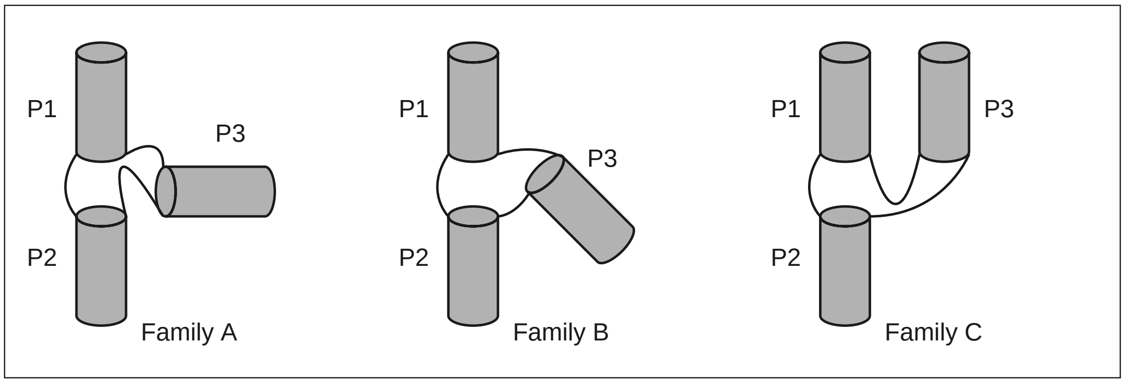


Figure 2.1 Three RNA topological families, A, B, and C, of 3-Way RNA junctions containing a coaxial helical stacking.

junction nearest to the 5' and 3' ends is generally considered to be the "first" helix. The number of a helix does not necessarily indicate its position relative to the 5' and 3' ends of the RNA molecule. In each 3-way RNA junction in Figure 2.1, the unpaired base region between P1 and P2 is called loop strand J12, the unpaired base region between P2 and P3 is called loop strand J23, and the unpaired base region between P3 and P1 is called loop strand J31.

The following are characteristics of each of the three topology families of 3-way RNA junction:

- In RNA topology family A:
  - loop strand J12 is the shortest of the three inter-helical loop strands;
  - loop strand J31 is typically shorter than loop strand J23;
  - P3 is roughly perpendicular to the coaxially stacking of P1 and P2.
- In RNA topology family B:
  - the three loop strands J12, J23 and J31, are all approximately the same length;
  - P3 is oriented closer to P2 than to P1.
- In RNA topology family C:
  - loop strand J12 is the shortest of the three inter-helical loop strands;
  - loop strand J31 is typically longer than loop strand J23;
  - P3 is oriented closer to P1 than to P2.

It is believed that the function of RNA is closely associated with its 3D structure, which, by virtue of canonical Watson-Crick base pairings (i.e., AU, GC) and wobble base pairing (i.e., GU), is largely determined by its secondary structure [66, 90, 92]. Several tools are available for secondary structure prediction and ncRNA search. One of the most highly regarded of these tools is Infernal [82], discussed in this chapter's previous case study. A wide variety of statistical analysis approaches, in particular, ensemble-based approaches, have been successful in life science applications. Laing, *et al.*, applied an ensemble-based approach, specifically random forests, to predict the existence of a coaxial helical stacking in RNA junctions [62].

In this case study, the functionality of Infernal was extended to create the CSminer tool to predict the existence of a tertiary RNA motif, i.e., a coaxial helical stacking, in a genome. This is accomplished by invoking a random forests classifier within Infernal to evaluate each significantly high-scoring Infernal search result and report the coaxial stacking status of these results. The secondary structure in each Infernal search result is formatted into Connectivity Table (CT) format and evaluated by a random forests classifier to confirm the pattern match.

2.3.2 Methods

Laing, *et al.*, studied 110 distinct 3-way RNA junctions that were confirmed in crystal structures [62]. Each of these 110 unique junctions was verified in one of 32 crystal structure models in the Protein Data Bank (PDB) [12]. The majority, 75%, of these 110 3-way RNA junctions were found in the complex ribosome subunit molecules (rRNA), i.e., 51% in 23S rRNA, 20% in 16S rRNA and 4% in 5S rRNA. Kingdoms represented in these 32 PDB samples were bacteria, archaea, animalia and plantae.

There was no dominant topological configuration among these 110 3-way RNA junctions in that 47% were categorized as family C, 35% as family A and the remaining 18% as family B [62]. For each of these 110 3-way RNA junctions, the coaxial helical stacking status was known. The coaxial stacking status of each 3-way RNA junction was described as one of four possibilities: H_1H_2, H_1H_3, H_2H_3 or none, where H_xH_y indicated that helix H_x shared a common axis with helix H_y. The helix identified as H_1 was the "first" helix in the 3-way RNA junction, as described below.

A 3-way RNA junction is described by three subsequences [62]. For each subsequence, base coordinates and base values (i.e., A, C, G, U) are known. The starting and ending coordinates of each subsequence indicate the 5' and 3' ends of the subsequence, respectively. Unpaired bases of each subsequence are referred to as part of the "loop regions" of the junction and are used to help determine the coaxial helical stacking status of the junction as described later. The 3-way RNA junction formed by these three subsequences includes unpaired bases of the loop region, terminal base pairs of the three helices and the second to last base pairs of the three helices, as follows. The 5' end of the first subsequence is the 5' base of the second to last base pair of helix H_1. The 3' end of the first subsequence is the 5' base of the second to last base pair of helix H_2. Similarly, the 5' end of the second subsequence is the 3' base of the second to last base pair of helix H_2, and the 3' end of the second subsequence is the 5' base of the second to last base pair of helix H_3. It follows that the 5' end of the third subsequence is the 3' base of the second to last base pair of helix H_3, and the 3' end of the third subsequence is the 3' base of the second to last base pair of helix H_1. The length of each subsequence is at least 4. The first two bases of each subsequence are part of one helix and the last two bases of that subsequence are part of the next sequential helix. There are zero or more unpaired bases between the two helices that share a subsequence.

Figures 2.2 and 2.3 illustrate a 3-way RNA junction in nucleotide positions 5 through 49 of chain A in PDB molecule 3E5C, i.e., "Crystal Structure of the SMK box (SAM-III) Riboswitch with SAM." This 3-way RNA junction is known to have a coaxial helical stacking identified as H_2H_3, i.e., helices H_2 and H_3 share a common axis.

The secondary structure plot for the RNA sequence is shown in Figure 2.2, which was produced using VARNA [26]. In this figure, the 3-way RNA junction is enclosed within a dashed line. The first subsequence of the 3-way RNA junction starts at position 5, ends at position 10 and consists of the bases CCGAAA. The second subsequence of the 3-way RNA junction starts at position 34, ends at position 41 and consists of the bases UUGUAACC. Finally, the third subsequence of the 3-way RNA junction starts at position 46, ends at position 49 and consists of the bases GGGG. Unpaired bases in the loop region are those bases not part of the terminal base pairs of the three helices. In this figure, helices H_2 and H_3 are shown to be coaxially stacked with the aid of a super-imposed bar.

Figure 2.3 was obtained using Jmol [48]. This figure presents a three-dimensional representation of the same RNA molecule shown in Figure 2.2, i.e., positions 5 through 49 of chain A in PDB molecule 3E5C. This figure represents the crystal structure 3D coordinates of the 976 atoms that comprise this RNA molecule. In this illustration, helix H_1 base positions 5 and 6 are identified, as are helix H_2 positions 34 and 35, and helix H_3 positions 46 and 47. The coaxial helical stacking of H_2 and H_3 is apparent in this illustration. In addition, Jmol provides interactive viewing of 3D figure rotations. By rotating and viewing the figure from any angle, the coaxial helical stacking becomes more visible.

A coaxial helical stacking motif in a 3-way RNA junction can be predicted by a random forests classifier that has been trained using certain specifically chosen "features" readily available in the secondary structure of known 3-way RNA junctions, i.e., the 110 element dataset described above. Collecting appropriate features for motif prediction is among the difficult yet important challenges in bioinformatics, pattern recognition and machine learning. Features used for this case study were previously

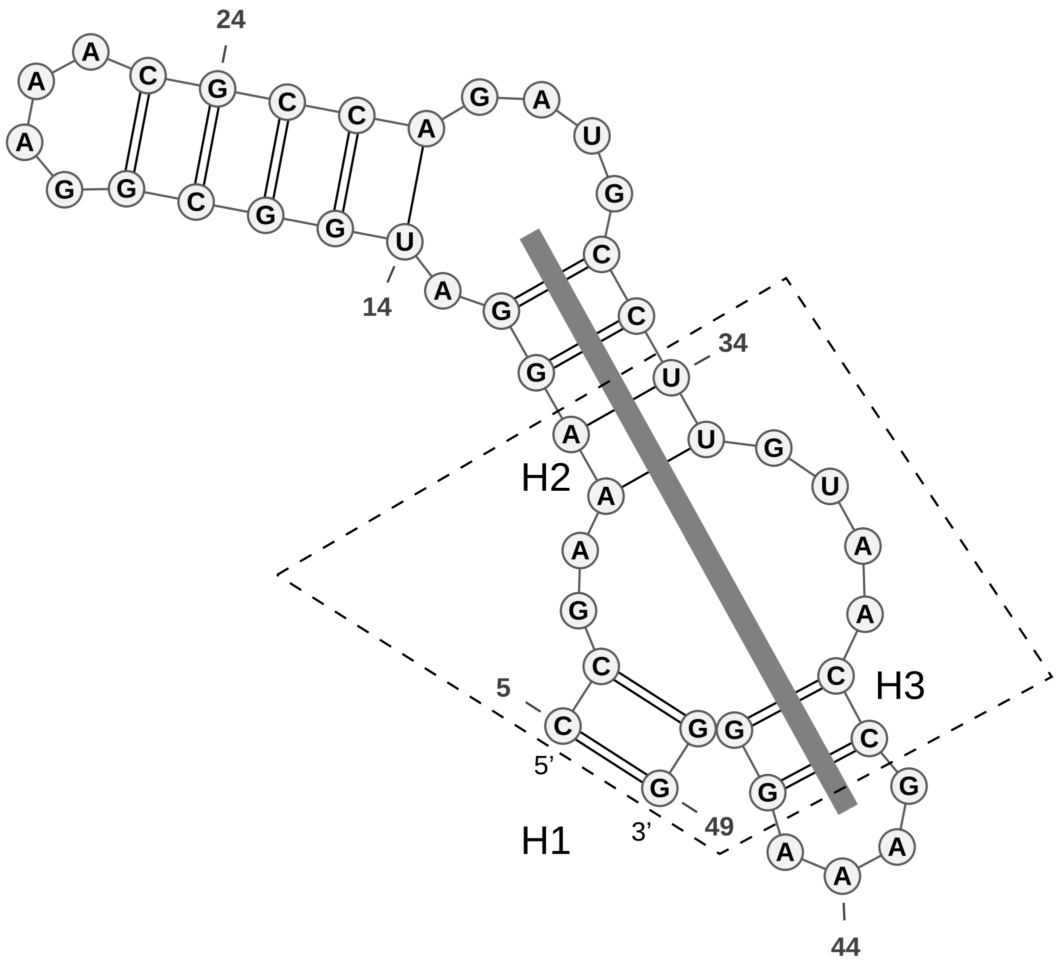


Figure 2.2 Secondary structure plot of chain A from PDB molecule 3E5C.

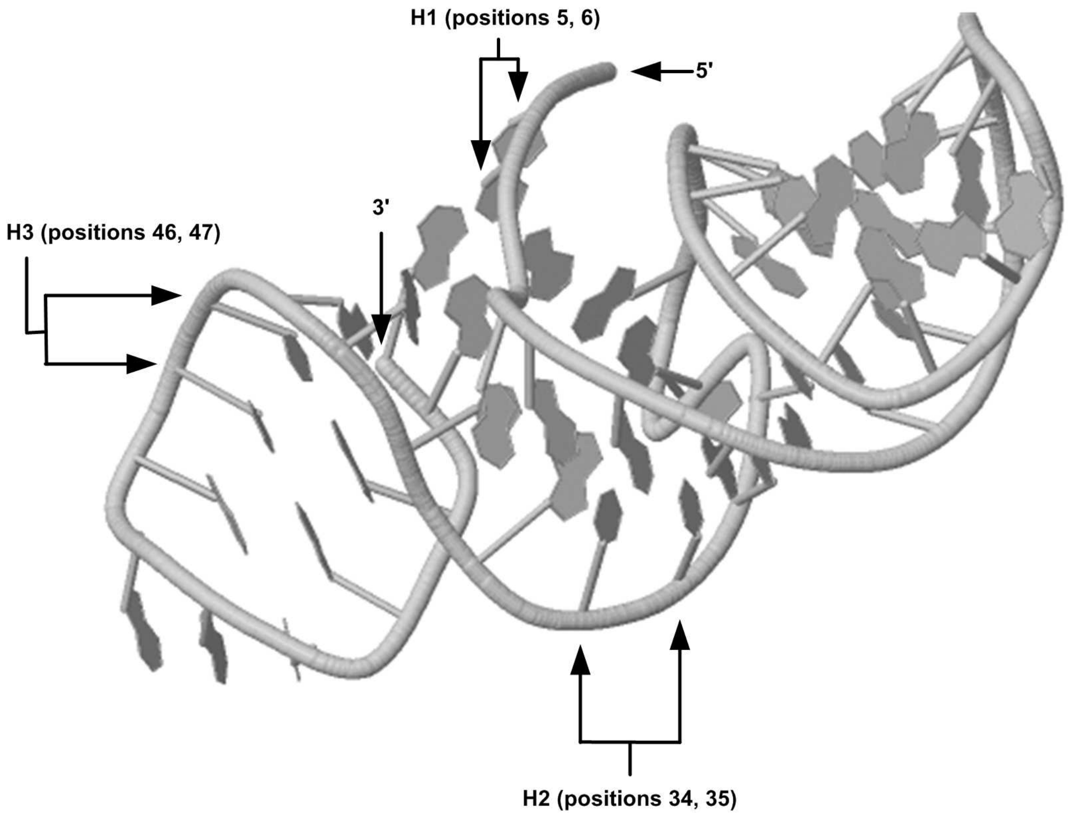


Figure 2.3 Three-dimensional plot of chain A from PDB molecule 3E5C.

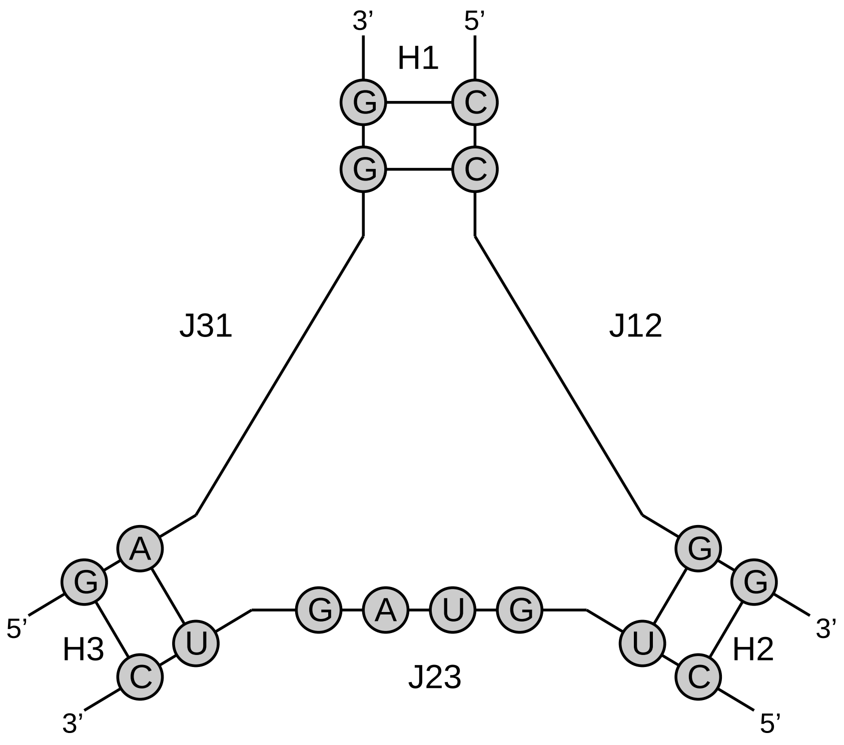


Figure 2.4 Hypothetical 3-way RNA junction illustrating random forests classifier features.

collected [62]. Figure 2.4 shows a hypothetical 3-way RNA junction that illustrates features used in this case study’s random forests classifier. Helix regions, consisting of paired nucleotides, are identified as H_1, H_2 and H_3. Loop regions, consisting of unpaired nucleotides, are identified as J12, J23 and J31.

Table 2.4 describes 15 features used to train the random forests classifier employed in the case study. Feature values were derived from attributes of known 3-way RNA junctions. A hypothetical 3-way RNA junction is shown in Figure 2.4. Features used are based on three principles.

- A short loop region in the 3-way RNA junction, i.e., the unpaired strand between adjacent helices, is more likely to be associated with a coaxial helical stacking. For this reason, the sizes or lengths of the three loop regions (i.e., the numbers of unpaired nucleotides in the three loop regions) of a 3-way RNA junction are used as features as well as the manner in which these three sizes relate to one another, e.g., the minimum, median and maximum of the three sizes.

- It is known that consecutive unpaired adenine bases tend to interact via hydrogen bonding with the minor groove of a neighboring helix. This common interaction, known as A-minor motif, stabilizes contacts between RNA helices. In fact, the A-minor motif is the most common tertiary interaction in the large ribosomal subunits. For this reason, information about consecutive unpaired adenine bases is used as features.
- Thermodynamic free energy associated with the base pairs at the helix termini and the loop regions between adjacent helices is used as features. It is known that as thermodynamic free energy declines in a conformation, stability increases.

In total, 15 features were used for coaxial helical stacking prediction to train the random forests classifier used in this case study.

Thermodynamic free energy of two adjacent helices was determined by the length or size of the unpaired nucleotide loop region between the two helices, denoted *LoopSize*, as follows:

- When *LoopSize* was 0, the free energy values were taken from the table of RNAstructure [93].
- When *LoopSize* was 1, the free energy values were taken from the table of RNAstructure, plus 2.1. If two possible values exist, the smaller one was taken.
- When *LoopSize* ranged from 2 to 6, the free energy values were calculated as follows: $a + b * LoopSize + c * h$ where a = 9.3, b = -0.3, c = -0.9, h = 2.
- When *LoopSize* was greater than 6, the free energy values were calculated as follows: $a + 6 * b + 1.1 * ln(LoopSize/6) + c * h$ where a = 9.3, b = -0.3, c = -0.9, h = 2.

The CSminer program combines the trained random forests classifier described above with Infernal [82], which was discussed in this chapter's previous case study. Complete nucleotide sequences were extracted, starting at the 5' end and ending at the 3' end, from the Protein Data Bank (PDB) [12] for all 110 known 3-way RNA junctions described above. Using RNAview [123] for guidance, the 110 secondary structures were manually evaluated. Out of the 110 secondary structures, 31 were selected based on similarity of length and general secondary structure. When searching a genome for ncRNA secondary structure motif matches, Infernal uses a covariance model (CM) comprised of several similar ncRNA secondary structures. As Infernal builds a CM, it takes into account the differences among the structures used in building the model,

Table 2.4 Features Used in the Random Forests Classifier for the Case Study Including Name, Value and Description for Each Classifier Feature

Feature	**Value**	**Description**
A(J12)	0	Adenine bases in loop region J12
A(J23)	1	Adenine bases in loop region J23
A(J31)	0	Adenine bases in loop region J31
ΔG(H_1,H_2)	-1.4	Thermodynamic free energy of helices H_1 and H_2
ΔG(H_2,H_3)	6.3	Thermodynamic free energy of helices H_2 and H_3
ΔG(H_1,H_3)	-2.1	Thermodynamic free energy of helices H_1 and H_3
|J12|	0	Length of J12 loop region in bases
|J23|	4	Length of J23 loop region in bases
|J31|	0	Length of J31 loop region in bases
Min(|J12|,|J23|,|J31|)	0	Minimum of 3 loop region lengths
Med(|J12|,|J23|,|J31|)	0	Median of 3 loop region lengths
Max(|J12|,|J23|,|J31|)	4	Maximum of 3 loop region lengths
Min(|J12|,|J31|)	0	Minimum of J12 and J31 loop region lengths
Min(|J12|,|J23|)	0	Minimum of J12 and J23 loop region lengths
Min(|J23|,|J31|)	0	Minimum of J23 and J31 loop region lengths

and shapes the model with statistical representations of these differences. Since the purpose of Infernal is to find structures similar to the CM, the structures that comprise the CM must also be similar. The more members that make up the CM, the more effective the model becomes. Even though a CM can be built using a single ncRNA structure, such a model would have significantly reduced effectiveness in locating similar structures.

The 31 selected secondary structures were clustered using RNAforester [49]. RNAforester clusters secondary structures based on secondary structure similarity. Six 3-way RNA junctions with similar secondary structures were grouped into a high-scoring cluster by RNAforester. These six 3-way RNA junctions had similar secondary structures and known coaxial helical stackings. The six 3-way RNA junctions belonged to PDB molecules with identifiers 2GDI, 2CKY, 2AVY, 1S72, 2AW4 and 2J01, respectively. These six 3-way RNA junctions formed the CM used in the case study.

A Stockholm format multiple sequence alignment is required to create an Infernal CM. The structure alignment provided by RNAforester was manually extracted, along with the consensus secondary structure. These established the required Stockholm format multiple sequence alignment (Figure 2.5). The Stockholm format multiple sequence alignment is a multiple alignment of ncRNA sequences together with the consensus secondary structure of the aligned sequences. The secondary structure is shown in dot-parentheses notation, in which dots represent bases and parentheses represent base pairs. The CM was created from the constructed Stockholm format multiple sequence alignment using Infernal's CMbuild utility [82].

Infernal's CMsearch utility was extended to execute a trained random forests classifier whenever an ncRNA secondary structure similar to the covariance model was detected during genome-wide searches performed by CMsearch. The resulting program was named CSminer.

```
# STOCKHOLM 1.0
#=GF ID 2GDI(94) (riboswitch)    length=74
#=GF ID 2CKY(96) A. thaliana     length=73
#=GF ID 2AVY(82) E. coli         length=71
#=GF ID 1S72(21) H. marismortui  length=73
#=GF ID 2AW4(22) E. coli         length=71
#=GF ID 2J01(23) T. thermophilus length=71
#=GF ID COAX_Model_19 (CM model name)
#=GF alignment and consensus structure by RNAforester software

2GDI_94.ct               CUCGGGGU----GCC-CUUCUGCGUGAAGGCUGAGAAAUACCCGUAUCACCU-GA
2CKY_96.ct               ACCAGGGG----UGC--UUGUUCAC-AGGCUGAGAAAGUCCCU-UUGAACCU-GA
2AVY_82.ct               UUAUCCUUUGUUGCCAGCGGUCCGGCCGGGAACUCA-A-AGGA--G--ACUG-C-
1S72_21.ct               GACAAGAUGAAGCG--UGCCGAAAG-GCACGUGG-A-AGUCUG--UU-AGAGUU-
2AW4_22.ct               GGCAGGUUGAAGGU--UGGGUAACA-CUAACUGG-A-GGACCG--A--ACCGAC-
2J01_23.ct               GCCAGGGUGAAGCU--GGGGUGAGA-CCCAGUGG-A-GGCCCG--A--ACCGGU-
#=GC SS_cons             ((.((((((((((((..(((.....)))))))).).)).)))).....((((.((

2GDI_94.ct               UC-UGGAUAAUGCCAGCGUAGGG-AA--G
2CKY_96.ct               AC-AGGGUAAUGCCUGCGCAGGG-AGUGU
2AVY_82.ct               CA-GUGAU--AA-ACUGGA-GGAAGGUGG
1S72_21.ct               GGUGUCCUACAAUACCCUC-UCG-UGAUC
2AW4_22.ct               UA-AUGUUGAAAAAUUAGC-GGA-UGACU
2J01_23.ct               GG-GGGAUGCAAACCCCUC-GGA-UGAGC
#=GC SS_cons             .(.(((......)))))).))).)...))
//
```

Figure 2.5 Stockholm format multiple sequence alignment of ncRNA molecules from six PDB samples.

The trained random forests classifier was capable of predicting a coaxial helical stacking in a 3-way RNA junction within the ncRNA secondary structure detected by CMsearch. Breiman designed the random forests classifier to be comprised of numerous classification and regression trees (CARTs) [17], each of which is formed by a small random subset of 4 (i.e., the square root) of the 15 features. Each CART is capable of contributing a "better than random opinion" about the coaxial helical stacking prediction of an unknown or unlabeled input. By consolidating all opinions from all CARTs, i.e., by tallying all "votes", the random forests classifier is able to predict the coaxial helical stacking status of the 3-way RNA junction.

It takes constant time for the random forests classifier to make predictions, and the space used by the random forests classifier is independent of the genome length. Thus, the space and time complexities of CSminer are the same as Infernal. Specifically, the space complexity of CSminer is $O(L^2M)$ and the time complexity is $O(L^3M)$, where L is the genome length and M is the number of states in the stochastic context-free grammar represented by the covariance model (CM) [28]. "Wallclock" run

```
CM: COAX_Model_19
>gi|15805042|ref|NC_001263.1|

  Plus strand results:

 Query = 1 - 74, Target = 251047 - 251117
 Score = 48.18, GC = 62

 Coax status = H2H3

           ((,<<<<<-<<<<<<-<<______>>>>>>>->->->>>>,,,<<--<-<<<<______>
  1 gcCaGGguGGaGgcCuggGUacgaccGgcUgGCAagcCCgauACCGacuggugaUAAAAc 60
           :CCAGG::G+A :CC ::GU++ A::GG: GG A:G:CCGA ACCG :+ :UG U+AAAC
    251047 ACCAGGUUGAAACCC-CCGUGACAGGGGGCGG-AGGACCGA-ACCGGUGCCUGCUGAAAC 251103

    >>>->->>,,,,))
        61 acccgcGGguGagc 74
           A: C:CGG+UGAG:
    251104 AGUCUCGGAUGAGU 251117
//
```

Figure 2.6 CSminer's prediction result on the genome of *D. radiodurans*.

times varied from 1 second to 1 hour 21 minutes and 48 seconds depending on the size of the target genome and the length of the aligned structures in the CM.

2.3.3 Results

A series of experiments were conducted to evaluate the pattern discovery effectiveness of the approach presented in this case study. In the first experiment, CSminer was run against the complete genome of *Deinococcus radiodurans*, i.e., GenBank ID NC_001263.1, obtained from the NCBI GenBank database [11]. An ncRNA tertiary motif, i.e., H_2H_3 coaxial helical stacking, was detected between positions 251047 and 251117 on the plus strand of the genome (Figure 2.6). This 3-way RNA junction is predicted by CSminer to contain a coaxial helical stacking. The coaxial helical stacking is of type H_2H_3 (i.e., helix H_2 and helix H_3 are aligned with a common axis).

This CSminer prediction result is validated as follows. Based on NCBI BLAST [6] and manual analyses, it is known that *D. radiodurans* is related to PDB molecule 1NKW. Specifically, the chain 0 nucleotide sequence was downloaded from the PDB for the 1NKW structure. Using NCBI BLAST, this downloaded sequence was located in the whole genome of *D. radiodurans*, i.e., GenBank ID NC_001263.1, from positions

251047 through 251117 on the plus strand. These positions are consistent with those shown by CSminer where the motif was detected (Figure 2.6). Furthermore, based on previous analysis [62], this region of the 1NKW structure contains a 3-way RNA junction with a coaxial helical stacking of type H_2H_3, which is what CSminer reports.

Table 2.5 presents the successful search results from 12 different CSminer experiments using three different covariance models (CMs) where the CMs were built using the techniques described in the previous subsection. A search is considered successful when the Infernal CMsearch score is higher than 30. The table contains the following columns:

- "Model PDB ID's": This column shows the PDB molecules from which RNA sequences of known coaxial helical stacking ncRNA tertiary motifs were extracted, aligned and used to build a CM for the CSminer genome search. The source species from which the PDB molecules come can be found in Table 2.6.
- "Genome ID": This column shows the accession number representing the genome sequence searched by CSminer with the respective CM formed with RNA sequences extracted from the PDB ID's shown in the first column. Note that the genome searched with a CM is different from the genomes/species from which RNA sequences were extracted and used to build that CM.
- "Species": This column shows the name of the species corresponding to the Genome ID in the second column.
- "Positions/Strand": This column shows search result positions in the genome sequence where CSminer predicts the location of a coaxial helical stacking. This column also shows the DNA strand, positive or negative, to which the search result positions pertain. Note that positions increase from low to high for a positive strand search result, and the positions decrease from high to low for a negative strand search result.
- "Status": This column shows the motif type predicted (H_1H_2, H_1H_3 or H_2H_3), where H_xH_y indicates that helix H_x shares a common axis with helix H_y.
- "Validated": This column shows whether or not the predicted result described by the fourth and fifth columns is validated by known crystal structure evidence in the PDB database. Where there is no available crystal structure evidence (i.e., when the column shows "no"), the predicted result needs to be validated by wet lab experiments.

CSminer was applied to the complete genome of *Thermus thermophilus*, i.e., GenBank ID CP002777.1. An ncRNA tertiary motif, i.e., H_1H_2 coaxial helical stacking, was detected between positions 14310 and 14384 on the plus strand of the genome. This result was validated by cross-checking the result coordinates against

Table 2.5 Successful Search Results from Twelve CSminer Experiments Showing Genome ID, Species, Positions, Strand, Coaxial Stacking Status and Validation Status for Each Result

Model	**Genome**	**Species**	**Positions**	**Strand**	**Status**	**Validated?**
1NKW,1S72,2Aw4	CP002777.1	*T. thermophilus*	14310-14384	+	H_1H_2	yes
	NC_013209.1	*A. pasteurianus*	1536843-1536769	-	H_1H_2	no
	NC_009484.1	*A. cryptum*	2585998-2585924	-	H_1H_2	no
	NC_016582.1	*S. bingchenggensis*	9707198-9707272	+	H_1H_2	no
1S72,2AVY,2AW4,	NC_001263.1	*D. radiodurans*	251047-251117	+	H_2H_3	yes
2CKY,2GDI,2J01	NC_013209.1	*A. pasteurianus*	1731899-1731829	-	H_2H_3	no
	NC_009484.1	*A. cryptum*	2009814-2009744	-	H_2H_3	no
	NC_016582.1	*S. bingchenggensis*	7289052-7289122	+	H_2H_3	no
1NKW,2AW4,2J01	NC_006397.1	*H. marismortui*	2771-2656	-	H_1H_2	yes
					H_2H_3	
	NC_013209.1	*A. pasteurianus*	1538830-1538717	-	H_1H_2	no
					H_2H_3	
	NC_009484.1	*A. cryptum*	2587983-2587870	-	H_1H_2	no
					H_2H_3	
	NC_016582.1	*S. bingchenggensis*	9704954-9705067	+	H_1H_2	no
					H_2H_3	

Table 2.6 Species and Kingdom for Each PDB Molecule Used to Build the Covariance Models Employed by CSminer

PDB ID	Species	Kingdom
2CKY	*Arabidopsis thaliana*	plantae
1NKW	*Deinococcus radiodurans*	bacteria
2AVY	*Escherichia coli*	bacteria
2AW4	*Escherichia coli*	bacteria
2GDI	*Escherichia coli*	bacteria
1S72	*Haloarcula marismortui*	archaea
2J01	*Thermus thermophilus*	bacteria

the known motifs in PDB molecule 2J01. This result is listed as "Validated" in Table 2.5.

CSminer was also applied to the complete genome of *Haloarcula marismortui*, i.e., GenBank ID NC_006397.1. Two ncRNA tertiary motifs, i.e., H_1H_2 and H_2H_3 coaxial helical stackings, were detected between positions 2771 and 2656 on the minus strand of the genome. This result was validated by cross-checking the result coordinates against the known motifs in PDB molecule 1S72. This result is also listed as "Validated" in Table 2.5.

In addition, experiments were conducted by selecting three bacterial genomes that are closely related phylogenetically to species represented in the covariance models (CMs) used in this case study. The three bacterial genomes selected were *Acetobacter pasteurianus* (GenBank ID NC_013209.1), *Acidiphilium cryptum* (GenBank ID NC_009484.1) and *Streptomyces bingchenggensis* (GenBank ID NC_016582.1). Experiments were also conducted to see whether these CMs would produce any meaningful search results in other biological kingdoms. Genomes selected were two viral genomes, one animal genome, one fungi genome and one protista genome. The two viral genomes selected were Human immunodeficiency virus 1

(GenBank ID NC_001802.1) and Human immunodeficiency virus 2 (GenBank ID NC_001722.1). The animal genome selected was *Drosophila melanogaster* chromosome X (GenBank ID NC_004354.3). The protista species chosen, *Plasmodium falciparum* (GenBank ID NC_004317), is a protozoan parasite and one of the species of *Plasmodium* that causes malaria in humans. The fungi species chosen, *Saccharomyces cerevisiae* (GenBank ID NC_001136), is one of the most intensively studied eukaryotic model organisms in molecular and cell biology, much like Escherichia coli as the model bacterium.

This ncRNA tertiary motif prediction method was performed on these additional genomes using CSminer. For each of the additional bacterial genomes, a motif was predicted. None of the additional bacteria organisms selected is represented in the PDB. Therefore, it cannot be confirmed that these predicted results are in fact coaxial helical stackings. These predictions are left to be validated with wet lab experiments. All the predicted results are listed in Table 2.5.

For the two viral genomes, the animal genome, the fungus genome and the protista genome, no motif was predicted. This was likely an indication that the motifs of interest were specific to the genomes/species represented in the covariance models (CMs) used by CSminer (cf. Table 2.6).

Two segments from PDB molecules 2GDI and 2CKY (used in the third CM in Table 2.5) are members of RFAM family RF00059, i.e., the "TPP riboswitch, also known as the THI element and Thi-box riboswitch" [41]. However, none of the results, either validated or non-validated, were found to overlap with a known RNA gene that is a member of some RFAM family. This is not unusual. When CSminer finds a match to the CM used in its search, that match need only be similar to a small number of structures comprising the CM. The match need not be similar to every structure comprising the CM.

2.3.4 Conclusion

This case study demonstrated that CSminer, by combining the strengths of the genome-wide ncRNA search tool, Infernal, with an ensemble-based random forests classifier, was an effective biological data mining instrument. Among the growing number of ensemble-based methodologies, the random forests method is among the most accurate. This functionality adds significant additional functionality to Infernal. Effective mining of coaxial helical stacking motifs in genomes will help to further unravel the mysteries of non-coding RNA. Much remains unknown in this exciting research area. This case study conclusion was that genome-wide mining of coaxial helical stacking RNA motifs was feasible and cost effective.

CHAPTER 3

BIOLOGICAL NETWORK INFERENCE: A COMPARATIVE REVIEW

3.1 Introduction

In life sciences research, the inference of gene regulation mechanisms in the cell is among the most active and interesting areas. Reverse engineering is the process of discovering the dynamic behavior and connectivity structure of a system given observations of the system. In this work, we attempt to identify the topology of a biological network through reverse engineering inference from experimental (ESCAPE [120]) and simulated (DREAM4 [75]) time-series gene expression data using publicly available implementations of time-series GRN inference tools. Understanding the topology of a GRN helps provide insights into the biology of cellular systems and potential targets of pharmacological compounds. Computationally reverse engineered GRNs help to simplify the daunting genetic analysis wet-lab process by drastically reducing the number of potential molecular interactions or locations of interaction sites to be investigated.

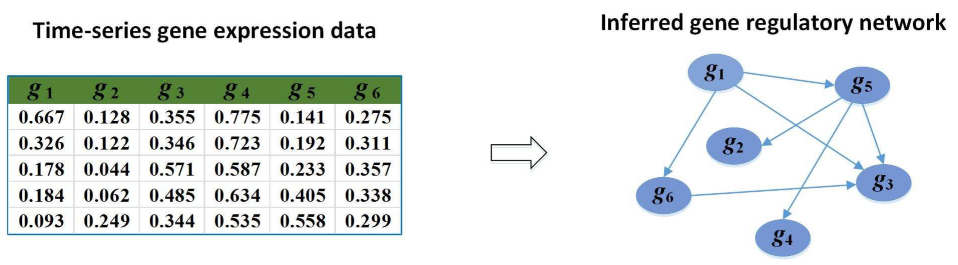
Advances in molecular biology, such as Next Generation Sequencing (NGS), allow researchers to the greatest extent ever possible to explore how genes regulate one another with nucleotide precision. There is much interest in computational approaches to reverse engineering genetic network probabilistic models from steady-state and time-series gene expression data [3, 69, 87]. Time-series data gathered in a microarray or RNA-Seq experiment consist of gene expression values recorded at each of a number of time intervals. Reverse engineering a GRN from digitized time-series gene expression data requires determining, for each (target) gene, the gene or genes most likely, i.e., most probable, to be regulators. A realistic GRN model helps guide effective disease treatment and intervention techniques through errant gene regulation correction. Once a GRN model is predicted through software, it must be translated into a hypothesis to be verified in a wet-lab experiment.

GRN inference from time-series gene expression data is challenging. Time-series gene expression data is typically sparse in the sense that the data contain many more variables (genes) than observations (experiments). That is, the number of observation time points (T) is usually much smaller than the number of genes (n) in time-series gene expression data sets. GRN inference algorithms take this situation into account to minimize false predictions when n is significantly larger than T.

NGS technologies simultaneously measure all gene expressions for even complex organisms containing tens of thousands of genes. GRN inference through reverse engineering, however, requires preprocessing to help reduce computational complexity. During a given experiment, for instance, many genes do not change their expression levels and are therefore less relevant to the experiment. After removal of irrelevant genes from the input, the regulatory relationships among the remaining genes are studied. Depending on the complexity of the algorithm, genes having similar expression patterns may be clustered and studied as a group [30] helping to further improve computational performance.

Here we present a survey of GRN time-series inference tools that use a variety of approaches. Each tool is applied to the same two time-series gene expression data sets: one is a simulated data set and the other is an experimental data set derived from wet lab experiments (see METHODS). Our objective is to provide a comparative overview of GRN time-series inference tool capabilities.

In a NGS time-series experiment, high throughput sequencing of the transcriptome occurs at specified time intervals typically following a perturbation intended to increase gene expression activity. Consecutive transcriptome sequencing reveals potential regulatory relationships among genes. For instance, plentiful expression of one gene shortly after plentiful expression of a different gene may be


Time-series gene expression data

g_1	g_2	g_3	g_4	g_5	g_6
0.667	0.128	0.355	0.775	0.141	0.275
0.326	0.122	0.346	0.723	0.192	0.311
0.178	0.044	0.571	0.587	0.233	0.357
0.184	0.062	0.485	0.634	0.405	0.338
0.093	0.249	0.344	0.535	0.558	0.299

Figure 3.1 Using a reverse engineering approach, a gene regulatory network (GRN) is inferred from time-series gene expression data produced from a microarray or RNA-Seq experiment.

evidence (to be evaluated) of an expression dependency, or regulation, between the two genes.

Figure 3.1 presents a high level overview of GRN inference of time-series gene expression data. On the left side of the figure, a table named "Time-series gene expression data" represents hypothetical results of a time-series gene expression experiment. This hypothetical experiment consists of measuring the expression values of six genes, named g_1 through g_6, over the course of five time points. The expression levels, i.e., observations, of the six genes for one time point comprise one row of the table, such that observations during the first time point comprise the first table row, observations during the second time point comprise the second table row, etc. It can be seen, for instance, that the expression value for gene $g6$ during the third time point is 0.357.

On the right side of Figure 3.1 is a directed graph named "Inferred gene regulatory network", which contains six nodes and seven edges. Each node represents one of the six genes, g_1 through g_6, from the gene expression table on the figure. A directed edge from one gene (a regulator) to another gene (a target) on the graph depicts a regulatory effect believed to exist between the two genes. For instance, the directed edge from regulator gene g_6 to target gene g_3 is meant to convey that gene g_3 is a target gene whose expression level is directly affected by gene g_6, one of g_3's regulator genes.

Researchers apply a perturbation, or unusual external influence, to cells during gene expression experiments with the intention of evoke a differential expression, i.e., a significant change in normal expression for a gene. A perturbation might include cell starvation, injury, chemical induction, extreme temperature shift, etc. Different perturbations evoke different cellular responses from different genes. Certain genes differentially expressed after perturbation of the immune system by influenza infection, for instance, may not be the same as those expressed following injury or starvation perturbation. Evolution of the GRN itself is studied in theoretical systems biology [64].

In spite of recent tremendous advances in GRN inference tool research, the problem of predicting a realistic remains elusive. Improved performance has been achieved, however, by combining GRN edge predictions from multiple methods [5, 31, 71, 74, 89]. By combining individual inference method results to form a community or group prediction for improved performance motivates us to use the frequent pattern mining technique described in this paper.

Here, we survey the performance of seven time-series GRN inference tools using simulated and experimental data sets described in METHODS. We find that, while each tool's performance might be considered unrealistic relative to the known gold standard for each respective network, contributions of all tools combined can be mined for frequent patterns resulting in a group performance that exceeds the individual performances of all tools taken individually. We conclude that group performance has significant potential for improving performance in time-series GRN inference research.

3.2 Methods

Two publicly available time-series gene expression data sets were downloaded for the experiments conducted in this survey. One data set was synthetic and the other was experimental.

A synthetic data set was downloaded from the DREAM (Dialogue for Reverse Engineering Assessments and Methods) initiative website. The DREAM initiative organizes annual reverse engineering competitions called the DREAM challenges [73]. In the DREAM4 edition, one challenge involved *in silico* regulatory network inference from synthetically generated gene expression data sets. The DREAM4 challenge was divided into three subchallenges named *In Silico* Size 10, *In Silico* Size 100, and *In Silico* Size 100 Multifactorial, where "Size" referred to the number of genes in the network. We downloaded our synthetic data set from the *In Silico* Size 10 subchallenge. This download included a gold standard network consisting of 15 gene interactions from which gene expression data sets were generated synthetically for DREAM4 challenges [40].

An experimental data set was obtained from the Embryonic Stem Cell Atlas from Pluripotency Evidence (ESCAPE) database [120]. A gene regulatory network (GRN) consisting of 30 genes was identified experimentally by Avi Ma'ayan's team [121]. The original chip_x network was downloaded from the Ma'ayan Lab ESCAPE website in a MySql database table format. There were 206,521 records and 178,841 unduplicated edges in the MySql database table format of the original chip_x network. Using the proposed 30 ground truth network genes as a reference, 84 non-self-directed edges were extracted from the MySql database table format of the original chip_x network to compile our proposed ESCAPE ground truth network.

Seven publicly available time-series GRN inference tools were downloaded and applied to the synthetic and experimental time-series gene expression data sets downloaded as described above. Six different GRN inference methods evaluated in this study were Bayesian, Boolean, tree-base, Granger causality (GC), information theoretic and ordinary differential equations (ODE).

3.2.1 Information Theoretic

Information theoretic models are based on the statistical analysis of dependencies between pairs of gene expression patterns (see Eq. (3.1)). A variety of popular mutual information (MI) tools has been developed, including ARACNE (Algorithm for the Reconstruction of Accurate Cellular Networks) [76] and TimeDelay-ARACNE [130]. The ARACNE algorithm, deployed in TimeDelay-ARACNE algorithm, filters out noisy indirect interactions among triplets of genes using Data Processing Inequality (DPI) [76, 130].

$$I^k(g_x, g_y^{(k)}) = \sum_{1 \leq i \leq n-k} p(g_x^i, g_y^{i+k}) \log \frac{p(g_x^i, g_y^{i+k})}{p(g_x^i)p(g_y^{i+k})} \tag{3.1}$$

For each pair of genes (g_x, g_y), time-delayed MI is computed as in Eq. (3.1) where n is the number of time points, g_x^i represents the gene expression at time point i, k denotes a time point interval, $p(g_x^i)$ is the marginal distribution of g_x^i and $p(g_x^i, g_y^{i+k})$ is the joint distribution of g_x^i and g_y^{i+k} [3].

ARACNE [76] first used the Data Processing Inequality (DPI) filter to remove indirect interactions between genes and reduce incorrect regulation predictions. According to DPI, if gene X_i interacts with gene X_k via gene X_j, and there is no other path from gene X_i to gene X_k, then the following inequality (Eq. (3.2)) must apply:

$$I(X_i, X_k) \leq (I(X_i, X_j), I(X_j, X_k)) \tag{3.2}$$

ARACNE considers the 3 pairwise MIs (i.e., 3 edges) in all instances of interactions among 3 genes and removes the smallest edge among the 3 if it falls below a given DPI tolerance for MI. Interactions within an interaction triangle are

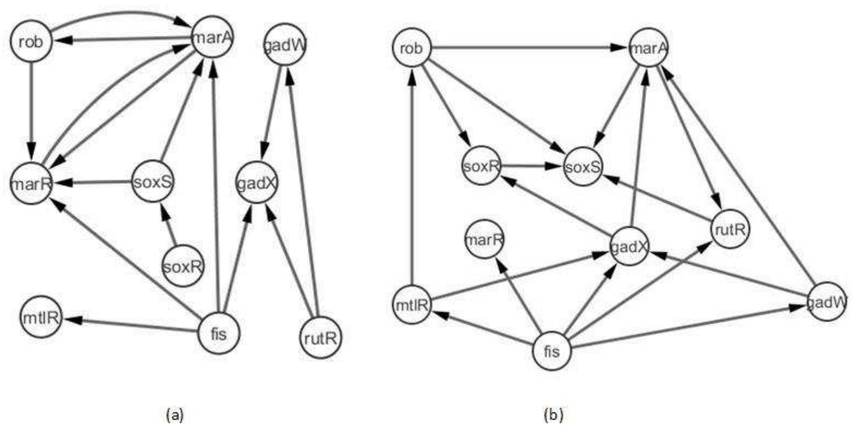


Figure 3.2 Comparison of: (a) DREAM4 gold standard gene regulatory network (GRN) and (b) TimeDelay-ARACNE's predicted GRN when applied to DREAM4 data. To aid comparative analysis, when a gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

taken to be indirect. Such indirect interactions are removed from the predicted GRN when they violate the DPI beyond a tolerance threshold.

TimeDelay-ARACNE version 1.20.0 was run using R version 3.2.5 on RStudio version 0.98.1103 on Windows 8.1 Pro. When TimeDelay-ARACNE was applied to DREAM4 data, it predicted a graph containing 15 edges. TimeDelay-ARACNE was applied to ESCAPE data and a total of 17 edges were predicted. Figure 3.2 illustrates how the GRN predicted by TimeDelay-ARACNE, when applied to DREAM4 data, compares with the DREAM4 gold standard GRN. Similarly, Figure 3.3 illustrates how the GRN predicted by TimeDelay-ARACNE, when applied to ESCAPE data, compares with the ESCAPE gold standard GRN. For easier comparative analysis of gene interactions in the figures, whenever the same gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

Another information theoretic tool evaluated for this survey was MIDER (Mutual Information Distance and Entropy Reduction) [111]. MIDER was

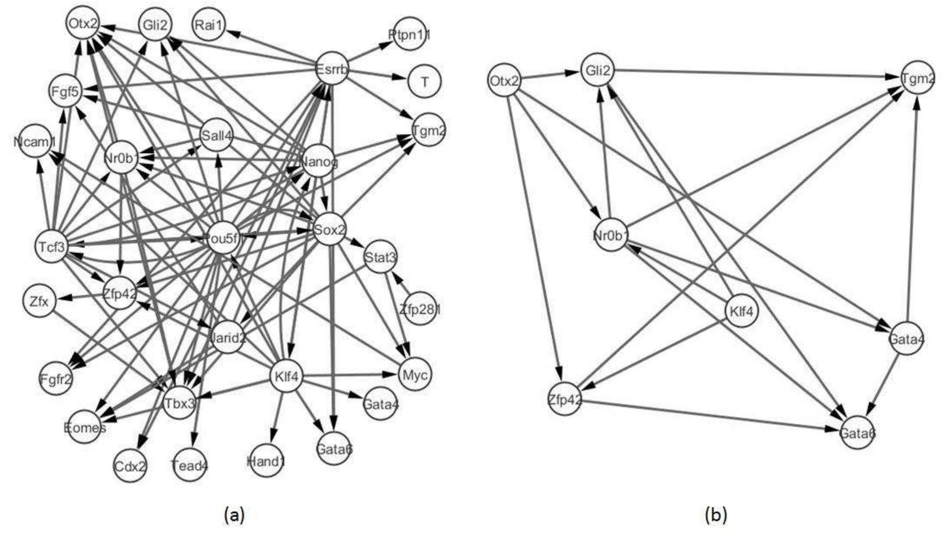


Figure 3.3 Comparison of: (a) ESCAPE gold standard gene regulatory network (GRN) and (b) TimeDelay-ARACNE's predicted GRN when applied to ESCAPE data. To aid comparative analysis, when a gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

downloaded from *http://www.iim.csic.es/ gingproc/mider.html* (last accessed on 4-3-2017). MIDER uses mutual information based entropic metrics and time delays to compute distances among variables (genes). This distance measure was used to determine regulatory effect between a pair of genes. MIDER version 2 was run using MATLAB version 9.1.0.441655 (R2016b) on Windows 8.1 Pro. The MIDER utility, *runMIDER.m*, was run and a GRN was produced. MIDER was able to infer a GRN for the DREAM4 time-series gene expression data. However, when attempting to apply MIDER to the ESCAPE time-series gene expression data, no GRN was produced. According to MIDER's author, Alex Villaverde, the problem appeared to be a lack of any calculable mutual information between pairs of variables at every time point. Therefore, MIDER results for the DREAM4 data set were reported, but there were no results to report for the ESCAPE data set. When MIDER was applied to DREAM4 data, it predicted a graph containing 13 edges.

Figure 3.4 illustrates how the GRN predicted by MIDER, when applied to DREAM4 data, compares with the DREAM4 gold standard GRN. For easier

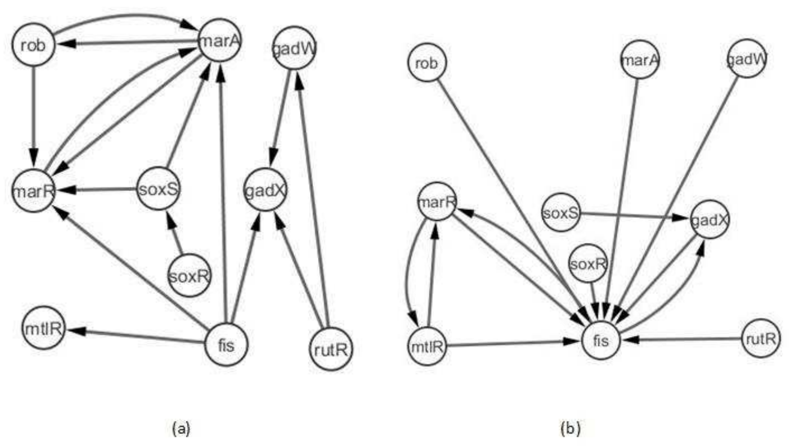


Figure 3.4 Comparison of: (a) DREAM4 gold standard gene regulatory network (GRN) and (b) MIDER's predicted GRN when applied to DREAM4 data. To aid comparative analysis, when a gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

comparative analysis of gene interactions in the figure, whenever the same gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

3.2.2 Bayesian

A Bayesian network (BN) is a probabilistic graphical model representing random variables and their conditional dependencies in a directed acyclic graph (DAG) [33]. BNs are used to represent GRNs inferred from steady state and time-series gene expression profile data. A steady state BN gene regulatory network (GRN) is a representation of a joint probability distribution in the form of a DAG, $G{=}(V, E)$, whose vertices, $\{V\}$, correspond to genes and whose edges, $\{E\}$, correspond to regulatory links between source genes and target genes in $\{V\}$. Each vertex $V_i \in V$ corresponds to a random variable X_i which can take on any specific gene expression value. The conditional distribution for each target gene is determined by its parents, i.e., source genes, that regulate it. The graph G follows the Markov property that each vertex (gene) is dependent only on its parents. The BN joint probability distribution of each variable $X_1, X_2, ..., X_p$ is defined as follows:

$$P(X_1, X_2, ..., X_p) = \prod_{i=1}^{p} P(X_i \mid \mathbf{Pa}^G(X_i)), \tag{3.3}$$

where p is the number of genes, $\mathbf{Pa}^G(X_i)$ is the set of parents for variable X_i as determined by the set of edges E in graph G, and $P(X_i|\mathbf{Pa}^G(X_i))$ is the conditional distribution for each variable, X_i.

A dynamic Bayesian network (DBN) is a Bayesian network relating variables to one another over adjacent time intervals [61]. In the case of a DBN, similar to the BN, the GRN represents a joint probability distribution in the form of a DAG, G=(V, E), whose vertices, $\{V\}$, correspond to genes and whose edges, $\{E\}$, correspond to regulatory links between source genes and target genes in $\{V\}$. In a time-series gene expression experiment, assume there are n time points at which expression levels of p genes are measured. The gene expression data can be represented by an $n \times p$ matrix X whose ith row contains expression values for p genes measured at time point i. The DBN joint probability distribution of each variable $X_{11}, ..., X_{1p}, X_{21}, ..., X_{np}$ is defined as follows:

$$\begin{aligned} &P(X_{11}, ..., X_{1p}, X_{21}, ..., X_{np}) \\ &\quad = P(\mathbf{X}_1) \times P(\mathbf{X}_2 \mid \mathbf{X}_1) \times ... \times P(\mathbf{X}_n \mid \mathbf{X}_{n-1}). \end{aligned} \tag{3.4}$$

where p is the number of genes, n is the number of time intervals and $\mathbf{X}_i = (X_{i1}, X_{i2}, ..., X_{ip})$ is a p-dimensional variable vector representing gene expression values of p genes at time interval i.

A Bayesian GRN inference tool estimates a probabilistic network of regulator-gene pairs from gene expression data. A Bayesian network (BN) defines the conditional probability distribution (CPD) of each child node given its parent nodes. Learning the structure of a Bayesian network is computationally intensive. Among the numerous approaches to perform a Bayesian network inference, the junction tree algorithm is among the most common [51, 54]. At each time interval, the objective is to estimate the parameters of each CPD to match the expression data. The ultimate goal is to create a model with the maximum likelihood, which is often a matter of maximizing the sum of the mutual information (MI) between each child node and its parent nodes [79]. A static Bayesian GRN network is acyclic in that it does not allow the presence of a cycle as a feedback loop. However, a cycle for a feedback loop is permitted by a dynamic Bayesian network model which is learned from time-series gene expression data. This approach performs reasonably quickly when the network size is small but becomes computationally prohibitive for medium to large size networks.

The networkBMA Bayesian-based time-series GRN inference tool was evaluated for this survey. To help compensate for sparse nature of time-series data, a Bayesian model averaging (BMA) approach can be used to approximate values between points in the time series and to eliminate genes that are highly correlated [126]. This approach is used in the networkBMA package evaluated in this study. networkBMA is an unsupervised GRN inference algorithm [127] that uses the Bayesian network inference computational approach and was developed within the R language and environment. networkBMA reads time-series gene expression profile data and predicts regulatory relationships between pairs of genes in a GRN. networkBMA addresses known limitations of Bayesian network inference, such as the exponential computational costs of evaluating networks that are not small [21, 22, 24]. A novel Bayesian model averaging (BMA) approach utilizes a more efficient model space search. networkBMA also performs time-series expression profile data transformations to improve performance in several areas, such as eliminating predictions of gene self-regulation.

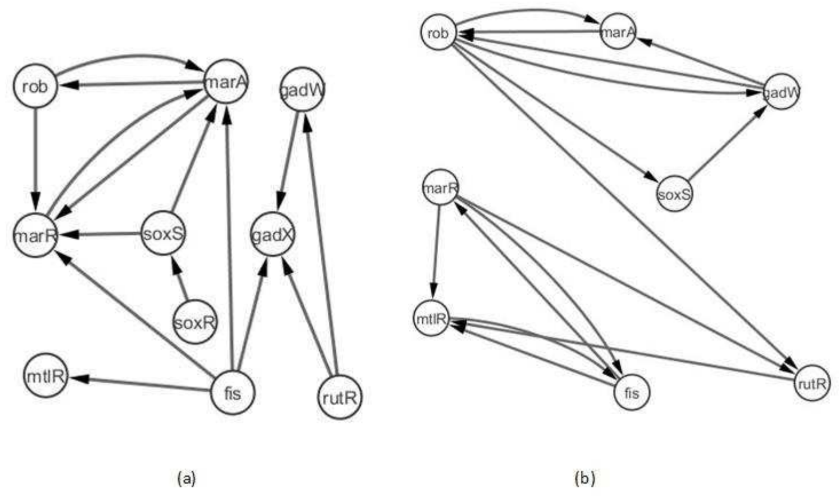


Figure 3.5 Comparison of: (a) DREAM4 gold standard gene regulatory network (GRN) and (b) networkBMA's predicted GRN when applied to DREAM4 data. To aid comparative analysis, when a gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

networkBMA version 1.12.0 was run using R version 3.2.5 on RStudio version 0.98.1103 on Windows 8.1 Pro. When networkBMA was applied to DREAM4 data, it predicted a graph containing 90 edges. The top 15 edges were chosen for performance evaluation per literature recommendation [64]. networkBMA was applied to ESCAPE data and a total of 870 edges were predicted. The top 84 edges were chosen for performance evaluation per the same literature recommendation. Figure 3.5 illustrates how the GRN predicted by networkBMA, when applied to DREAM4 data, compares with the DREAM4 gold standard GRN. Similarly, Figure 3.6 illustrates how the GRN predicted by networkBMA, when applied to ESCAPE data, compares with the ESCAPE gold standard GRN. For easier comparative analysis of gene interactions in the figures, whenever the same gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

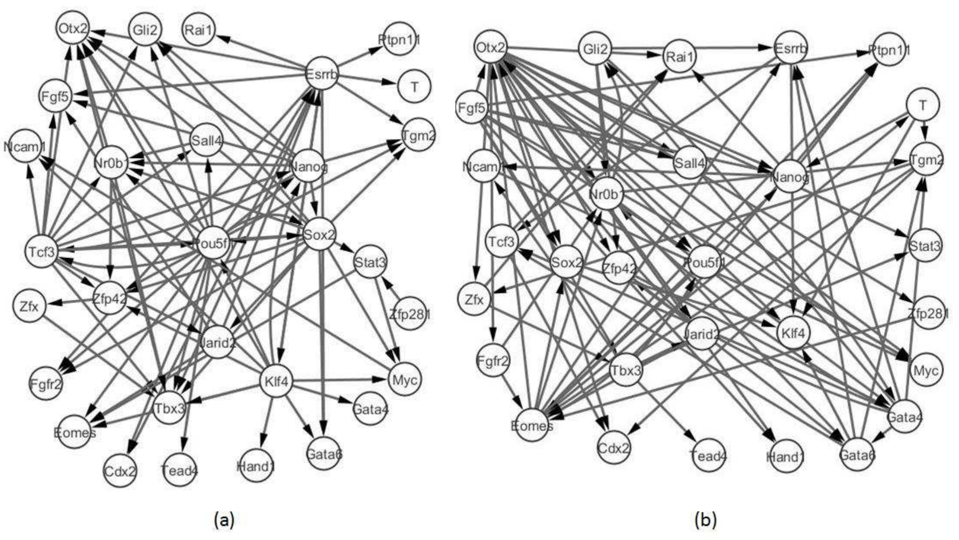


Figure 3.6 Comparison of: (a) ESCAPE gold standard gene regulatory network (GRN) and (b) networkBMA's predicted GRN when applied to ESCAPE data. To aid comparative analysis, when a gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

3.2.3 Granger Causality

Granger causality (also known as of Wiener-Granger Causality), occurs in a situation involving two time-series variables, X and Y, when a prediction is attempted for a subsequent value of X using only past values of X and a second prediction is attempted for a subsequent value of X using past values of both X and Y [39]. When the second prediction is significantly more successful than the first prediction, then Y is said to Granger-cause X. A Granger causal influence of one time-series on another time-series can be claimed if the prediction of one time-series can be significantly improved by using knowledge from the second time-series.

The Granger causality approach is common for inferring relationships among genes in time-series gene expression profiles. One gene, say g_2, is said to be causal for another gene, g_1, if the prediction of gene g_1 at time point t using all relevant information available at time point $t-1$ about gene g_1 can be significantly improved by also considering all relevant information available at time point $t-1$ about gene g_2.

An autoregressive model describes how a term at time point t in a gene expression time-series can be predicted from previous gene expression terms in the series at time points $t-1$, $t-2$, etc. Eq. (3.5) shows an autoregressive model prediction of the expression of a gene g_1 at time point t given the previous q terms in the gene expression time-series.

$$g_1^t = c + \alpha_1 g_1^{t-1} + ... + \alpha_q g_1^{t-q} + \epsilon_t \tag{3.5}$$

In the autoregressive Eq. (3.5), c is an initial constant, α_1 through α_q are coefficients of each previous gene g_1 expression in the series and ϵ_t is typical noise associated with gene expression profile experiment.

An autoregressive model is also used to determine if there is a Granger causality between two genes, e.g., g_1 and g_2. If the autoregressive model in Eq. (3.6) holds and at least one of the gene g_2 coefficient terms β_1 through β_q is not equal to zero, then there is a term in the g_2 time-series that contributes to the prediction of the expression value of gene g_1 at time point t, and gene g_2 is said to "Granger cause" gene g_1.

$$g_1^t = c + \alpha_1 g_1^{t-1} + ... + \alpha_q g_1^{t-q} + \beta_1 g_2^{t-1} + ... + \beta_q g_2^{t-q} + \epsilon_t \tag{3.6}$$

When time-series gene expression data for a higher number of genes, g_1 ... g_n, are available, the vector autoregressive (VAR) model is used for Granger causality prediction [44].

One of the primary goals of studying a time-series gene expression profile is to identify direct inter-gene causal relationships. [34, 78, 81]. Two of the most common methods often used to infer interactions among items in a time-series are the Granger causality method the Bayesian network inference method. Whereas the Bayesian method can be applied to static or time-series data, the Granger causality method is used only for time-series data. A direct causal relationship between two genes, i.e., $\{g_2 \rightarrow g_1\}$, implies that the expression of gene g_2 predicts the expression of gene g_1. Granger causality was originally developed for economic forecasting and has recently been successfully applied to the problem of gene regulatory network (GRN) inference [18, 131] using a GRN inference tool named CGC-2SPR.

The CGC-2PRS process for a hypothetical network was evaluated as follows as detailed in [124]. A hypothetical gene regulatory network was established containing 1,000 nodes and 1,082 regulatory edges. After running the CGC-2SPR (conditional Granger causality using two-step prior Ridge regularization) process using simulated gene expression data, the highest ranking edges were compared against the gold standard network from which the expression data were derived. The *BayesianRidge* algorithm produced regression matrix B using a lag time, p, of 3 to limit the range of the effect that one gene will have on another gene. The X matrix was comprised of 18 rows by 30 columns and represented three (i.e., $p = 3$) replicates of each of the 10 genes in the network for 18 (i.e., 21 - p) time-points in the time-series. The Y matrix was comprised of 18 rows by 10 columns and represented the original expression value for each of the 1,000 genes in the network for 18 (i.e., 21 - p) time-points in the time-series. The W matrix was comprised of 30 rows by 10 columns and represents prior knowledge about the regulatory network. Prior knowledge about an edge took the form of the value 1 in *ith* row and ((j-1) mod 1000 + 1)th column of the W matrix when a regulatory relationship from gene i to gene j was known to exist. (As previously stated, no prior knowledge was applied in tools used for this survey. as such, all values in matrix W were set to zero.) The regression matrix B was comprised of 30 rows by 10 columns representing scores of all pair-wise combinations of genes in the 10 gene network .

After computing regression matrix B, CGC-2SPR Perl and MATLAB scripts were run to score the results relative to the gold standard of the simulated network.

The Perl script named *analyse.pl* compared result edges with gold standard network edges and saved the edges that match. The Perl script named *prepare.pl* compared result edges with gold standard network edges and saved data recorded on the same line with the results (i.e., located in columns 3 and 4) recorded with an edge that matched a gold standard network edge. A predicted edge that matched a gold standard edge was known as a true positive (TP).

The CGC-2SPR GRN inference tool identified 3 true positives for the DREAM4 network. This is less than the 15 gold standard network edges obtained from the *In Silico* Size 10 subchallenge site [40].

The CGC-2PRS process described above was repeated for the ESCAPE experimental time-series gene expression profile. The CGC-2SPR GRN inference tool identified 18 true positives for the ESCAPE network. This is less than the 84 gold standard network edges obtained from the ESCAPE database [121, 120].

Note that CGC-2SPR was designed to be run using prior knowledge. For a fair comparison against other tools in this survey, CGC-2SPR was run without using prior knowledge since most tools surveyed did not have the ability to consider prior knowledge. Yet, in spite of the lack of prior knowledge, CGC-2SPR scored the highest among all tools in the GRN inference using the ESCAPE experimental time-series gene expression data set.

Modules of CGC-2SPR version 2015 were run using R version 3.2.5 on RStudio version 0.98.1103 on Windows 8.1 Pro. Other CGC-2SPR modules were run using MATLAB version R2015a (8.5.0.197613) 64-bit (glnxa64) on Linux version 2.6.32-642.6.2.el6.x86_64. Other CGC-2SPR modules were run using Perl version 5.10.1 on Linux version 2.6.32-642.6.2.el6.x86_64. When CGC-2SPR was applied to DREAM4 data, it predicted a graph containing 90 edges. The top 15 edges were chosen for performance evaluation per literature recommendation [64]. CGC-2SPR was applied to ESCAPE data and a total of 870 edges were predicted. The top 84 edges were

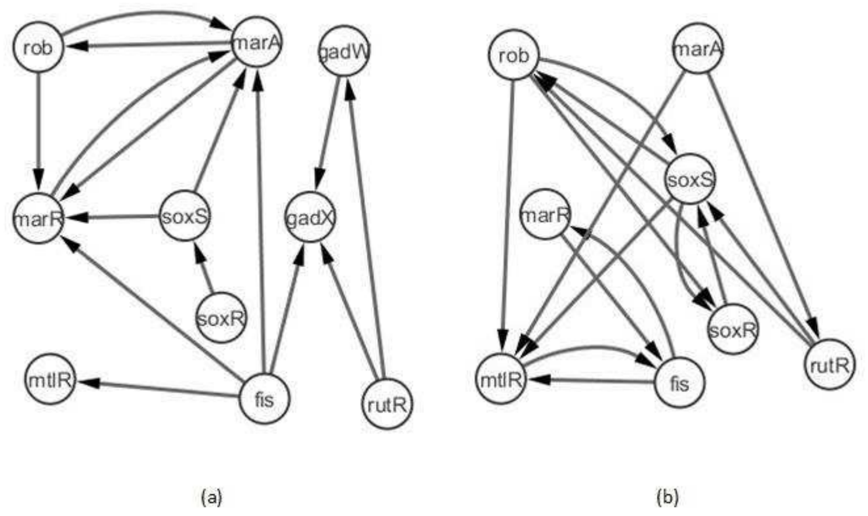


Figure 3.7 Comparison of: (a) DREAM4 gold standard gene regulatory network (GRN) and (b) CGC-2SPR's predicted GRN when applied to DREAM4 data. To aid comparative analysis, when a gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

chosen for performance evaluation per the same literature recommendation. Figure 3.7 illustrates how the GRN predicted by CGC-2SPR, when applied to DREAM4 data, compares with the DREAM4 gold standard GRN. Similarly, Figure 3.8 illustrates how the GRN predicted by CGC-2SPR, when applied to ESCAPE data, compares with the ESCAPE gold standard GRN. For easier comparative analysis of gene interactions in the figures, whenever the same gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

3.2.4 Boolean

A Boolean network (BLN), $B(G, R)$, is represented by a graph, B, consisting of nodes $G=\{g_1, g_2, ..., g_N\}$ representing N genes, and edges $R=\{(g_i \rightarrow g_j) | g_i, g_j \in R\}$ representing regulatory interactions between pairs of genes. Each node, g_i, represents a Boolean variable whose state is determined by a Boolean function of other Boolean

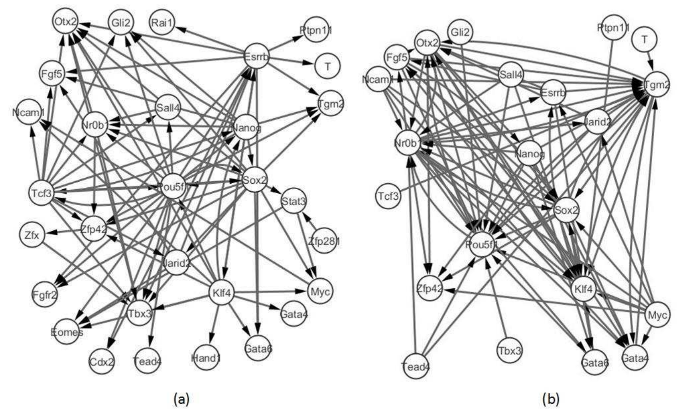


Figure 3.8 Comparison of: (a) ESCAPE gold standard gene regulatory network (GRN) and (b) CGC-2SPR's predicted GRN when applied to ESCAPE data. To aid comparative analysis, when a gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

variables. In particular, the state of node g_i at time $t+1$ results from K other nodes $g_{i_1}, g_{i_2}, ..., g_{i_K}$ having regulatory interactions on g_i (defined in R) at time t as described by the following Boolean function [10]:

$$g_i(t+1) = f_i(g_{i_1}(t), g_{i_2}(t), ..., g_{i_K}(t)). \tag{3.7}$$

Boolean networks (BLNs) and probabilistic Boolean networks (PBLNs) are highly studied mathematical models in GRN inference research [117]. BLNs and PBLNs represent the expression level of each gene as either *ON* or *OFF* (i.e., 1 and 0). This particular form of discretization for BLNs and PBLNs is called binarization. In this way, standard and well understood logic operations (AND, OR and NOT) are used to clearly describe gene regulatory interactions. The clarity of understanding provided by BLNs and PBLNs is often offset by information loss due to the binarization process. One of the advantages of a Boolean GRN is the ease of building the network with biological expert knowledge expressed in non-technical natural-language statements. Conversely, however, it's not clear whether a Boolean

GRN can be inferred from a pre-existing gene expression profile data set in the absence of such biological expert knowledge.

A BLN or PBLN model describes a GRN as transitioning among a finite number of states. A state in a BLN or PBLN is a binary vector of all the gene expression values (1 or 0) at any point in time, and the state space for a BLN and PBLN model consists of all the possible states, i.e., 2^n, for a model with n genes. Details for building the BN and PBN from a state transition model are described in [118].

Liang, et al, [68] proposed a deterministic process named REVEAL to identify an update rule for each gene of a network when the input and output values are known. This situation isn't feasible, however, where a reference network isn't available. Shmulevich, et al, [97] describes a *de novo* process to infer a probabilistic Boolean network without requiring a reference network.

Among the challenges when predicting a gene regulatory network using Boolean algebra methods is the discretization process. When a floating point gene expression value is converted to a binary integer, gene expression is deemed to be either on or off, like a simple electric switch. Valuable information may be lost in this prediction process. A misrepresented gene expression at one time point impacts predictions at subsequent time points since the predicted output for one gene is determined by the input of all genes in the network.

The BoolNet Boolean network based time-series GRN inference tool was evaluated for this survey. BoolNet [80] supports providing required edges and excluded edges as ”prior knowledge.” A Boolean network may be synchronous, asynchronous, probabilistic or temporal. BoolNet provides two algorithms for network reconstruction: REVEAL and Best-Fit Extension. BoolNet provides the ability to perform a heuristic search or an exhaustive search of a binarized gene expression data set.

Reconstruction of DREAM4 and ESCAPE data required normalization, or binarization of the gene expression data values. Data normalization can be based on difference of expression values or the average of expression values [15]. The BoolNet package has a built-in function that binarizes the gene expression data.

Customized R scripts *dm185.r* and *dm186.r* (available from the author upon request) parsed the raw output objects generated by BoolNet. These scripts formatted BoolNet predictions for performance evaluation and plotting tools. The BoolNet method named *reconstructNetwork* processed a network in matrix form and produced an object of R class *probabilisticBoolenNetwork,* which was printed using the BoolNet *print* method. The output of the BoolNet *print* method was parsed by customized R scripts (available from the author upon request) to extract desired edges for performance evaluation.

To evaluate the GRN inference performance of BoolNet, we formatted our DREAM4 and ESCAPE time-series data sets into R matrix objects required by BoolNet. In the required format, gene expression results for a specific point in time were arranged in columns and all expressions for a specific gene were recorded in a row. Gene identifiers were recorded in the first column prior to the gene expression values. There were no names for rows and columns, i.e., the method returned the value *NULL.* Boolean network inference requires the binarization of gene expression values, which results in the translation of each gene expression value into a *0* or *1.* The BoolNet method *binarizeTimeSeries* performed the binarization task and created an object of the same matrix class of the original time-series gene expression data set.

Let B be the matrix of binarized gene expression values. The BoolNet *reconstructNetwork* method predicted a probabilistic Boolean GRN comprised of optional regulatory edges $\{gene_1 \rightarrow gene_2, gene_1 \rightarrow gene_3, \dots , gene_1 \rightarrow gene_M\}$, $M < N$, for each of the N genes in the network, where, in each gene pair, the first gene is the regulatory gene and the second gene is the target gene. The $maxK$ parameter of the *reconstructNetwork* method limited the number of edges having a common regulating gene. Higher *maxK* parameter values consumed proportionately more CPU and memory computational resources. Prior gene regulatory interaction knowledge, when used, could be provided via the *excludeDependencies* and *requiredDependencies* parameters during the running of the *reconstructNetwork* method. The BoolNet *chooseNetwork* method produced one network by selecting edges within the probabilistic Boolean GRN produced by the *reonstructNetwork* method. The *chooseNetwork* method produced a list of transition functions from which GRN edges were extracted.

GRNs predicted by BoolNet varied depending on the order of genes in the input gene expression data set. This drawback was indicative of the steep complexity increase when larger networks were evaluated and was related to the need to specify parameters appropriately in order to obtain results in reasonable timeframes.

BoolNet version 2.1.1 was run using R version 3.2.5 on RStudio version 0.98.1103 on Windows 8.1 Pro. When BoolNet was applied to DREAM4 data, it predicted a graph containing 35 edges. The top 15 edges were chosen for performance evaluation per literature recommendation [64]. BoolNet was applied to ESCAPE data and a total of 66 edges were predicted. Figure 3.9 illustrates how the GRN predicted by BoolNet, when applied to DREAM4 data, compares with the DREAM4 gold standard GRN. Similarly, Figure 3.10 illustrates how the GRN predicted by BoolNet, when applied to ESCAPE data, compares with the ESCAPE gold standard GRN. For easier comparative analysis of gene interactions in the figures, whenever the same gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

3.2.5 Ordinary Differential Equations (ODE)

The inference of a gene regulatory network from time-series gene expression data can be performed using ordinary differential equations (ODE) whereby the instantaneous

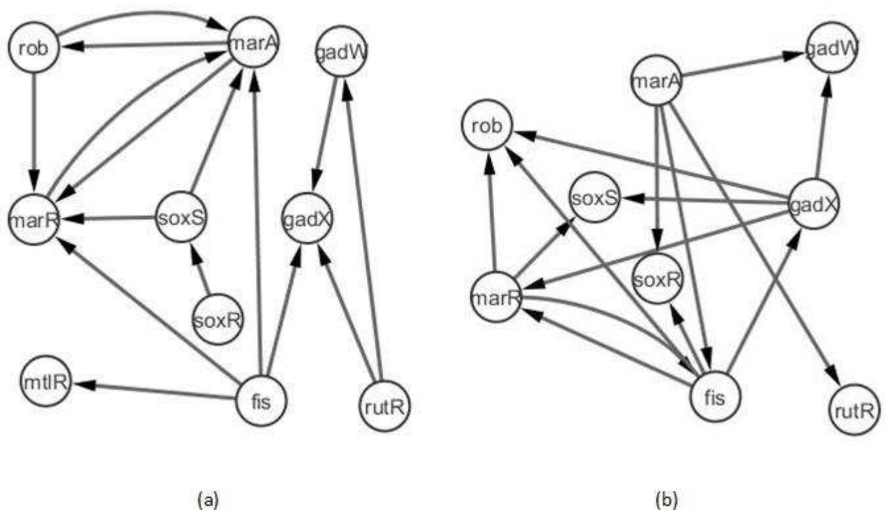


Figure 3.9 Comparison of: (a) DREAM4 gold standard gene regulatory network (GRN) and (b) BoolNet's predicted GRN when applied to DREAM4 data. To aid comparative analysis, when a gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

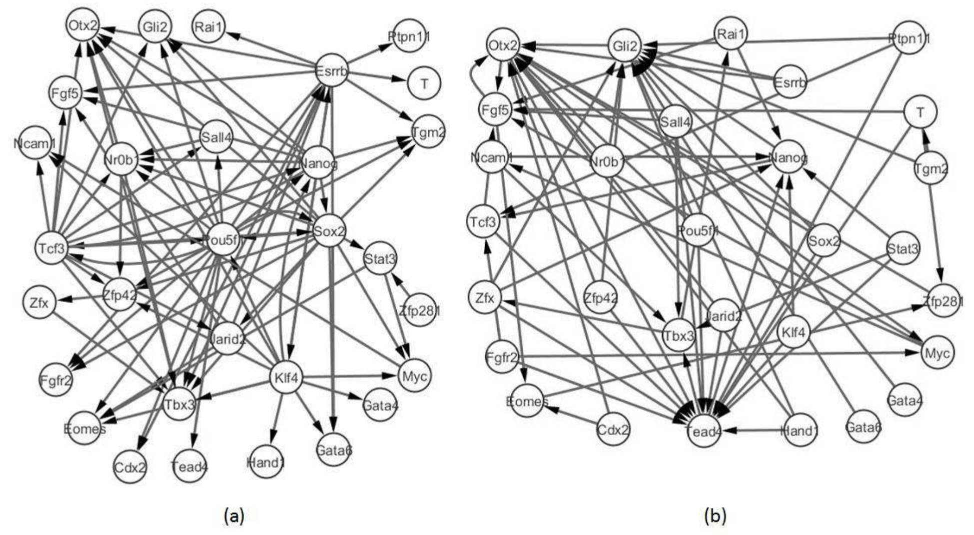


Figure 3.10 Comparison of: (a) ESCAPE gold standard gene regulatory network (GRN) and (b) BoolNet's predicted GRN when applied to ESCAPE data. To aid comparative analysis, when a gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

change in the expression level of a gene is expressed as a formulas involving expression level of other genes. Assuming that the behavior of a gene regulatory network (GRN) on N genes can be modeled by a system of nonlinear differential equations, the model is defined as an N x N matrix of coefficients describing the regulatory interactions between the genes [35].

In systems biology, the S-system model in Eq. (3.8) [94] is often used as a basis for inferring a GRN.

$$\frac{dx_i}{dt} = \alpha_i \prod_{j=1}^{N} x_j^{g_{i,j}} - \beta_i \prod_{j=1}^{N} x_j^{h_{i,j}}, \text{for } i = 1, 2, ..., N \tag{3.8}$$

In Eq. 3.8, parameters i and j each represent one network node, parameter g is a kinetic order representing the positive interaction between two network nodes, parameter h is a kinetic order representing the negative interaction between two network nodes, parameter α represents a non-negative constant of positive inter-node interaction for each network node, parameter β represents a non-negative constant of negative inter-node interaction for each network node and N represents the number of nodes in the network.

Where there is time-series information gathered about the output of each node of a network, the cause-and-effect relationship among the network nodes is believed to be derivable by solving for the parameters of the S-system that represents that network. This approach has considerable promise in the field of cellular biology research. A large and growing body of time-series gene expression profile data is available for evaluation. Where S-system parameters can be accurately derived regarding the genes involved in a time-series gene expression profile, researchers hope to glean critical information about cellular function, especially disease progression.

Efficiently solving for the best coefficients in the matrix is challenging. By defining a separate formula for each individual gene as a function of all other pertinent genes, the solution to the inference problem can be presumably determined by solving the problem of N simultaneous equations for N variables. While solving ordinary differential equations is less computationally intensive than solving partial differentially equation, the problem remains exponentially difficult, i.e., NP-hard, and requires heuristic approaches to avoid excessive run times for medium to large numbers of genes. The ODE methods considered here are applied to networks consisting of small numbers of genes.

Inferelator is a time-series GRN inference tool that applies gene groupings (where a group y is comprised of clusters of genes x) based on similarities in co-expression, and solves for the following ODE:

$$\frac{dy_i(t)}{dt} = \alpha_i y_i + \sum_{j=0}^{P} \beta_{i,j} f_j(x(t)), \text{ for } i = 1, 2, ..., N \tag{3.9}$$

where $\alpha_i y_i$ represents the degradation rate of y_i, β is a matrix of kinetic parameters to be computed, $f_j(x(t))$ represents the minimum of $x_i(t)$ or $x_{i'}(t)$ and N is the number of y groupings. Inferelator applies a Bayesian approach in combination with the above ODE method. Algorithm details are available in [72].

Inferelator is maintained at the GitHub project hosting site (*https://github.com/ChristophH/Inferelator*) [72]. Inferelator must be run in a non-Windows environment when multiple cores are deployed. Otherwise the message "'mc.cores' < 1 is not supported on Windows" is displayed. Sample jobs included with the download from GitHub, including jobs processing 100-gene DREAM4 networks, worked successful and served as good examples for setting up new jobs. Input data file directories for DREAM4 10-gene and the ECLIPSE 30-gene network were established and populated. Parse method options provided by Inferelator are BBSR and MEN. Inferelator has the capability to use prior knowledge to improve the confidence of GRN edge predictions. Inferelator was applied to DREAM4 and ESCAPE time-series gene expression data sets without the use of prior knowledge since prior knowledge information was not used when running other GRN inference tools.

Inferelator version 2.0 was run using R/Rscript version 3.2.4 on Linux version 2.6.32-642.6.2.el6.x86_64. When Inferelator was applied to DREAM4 data, it predicted a graph containing 42 edges. The top 15 edges were chosen for performance evaluation per literature recommendation [64]. Inferelator was applied to ESCAPE data and a total of 221 edges were predicted. The top 84 edges were chosen for performance evaluation per the same literature recommendation. Figure 3.11 illustrates how the GRN predicted by Inferelator, when applied to DREAM4 data, compares with the DREAM4 gold standard GRN. Similarly, Figure 3.12 illustrates how the GRN predicted by Inferelator, when applied to ESCAPE data, compares with the ESCAPE gold standard GRN. For easier comparative analysis of gene interactions in the figures, whenever the same gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

3.2.6 Tree-Based

Feature selection inference models leverage the tree-based approach for classification and regression. Random forests tree-based ensemble methods [16] decompose the problem of finding potential regulators of N genes into N distinct sub-problems. As each of the N genes is assumed to be a target gene, each of the N genes is ranked as a potential regulator gene using feature selection and random forests methods. Highest ranking genes are identified as potential regulators for the target gene in question. Using gene expression data, potential regulatory genes for a given target gene are identified as those genes whose expression influences the expression of the target gene.

Jump3 [53] enhances the tree-based ensemble approach used by GENIE3 [52] and uses the Extra-Trees procedure [36]. In this manner, rather than using a brute

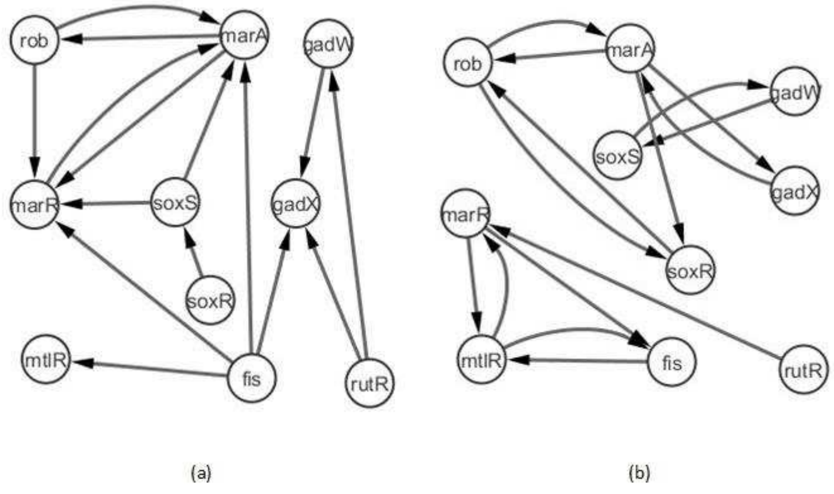


Figure 3.11 Comparison of: (a) DREAM4 gold standard gene regulatory network (GRN) and (b) Inferelator's predicted GRN when applied to DREAM4 data. To aid comparative analysis, when a gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

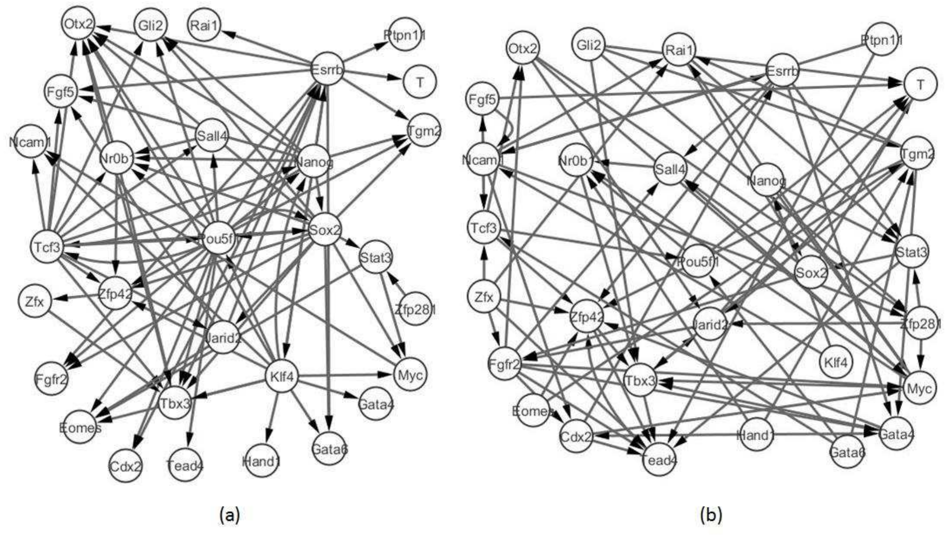


Figure 3.12 Comparison of: (a) ESCAPE gold standard gene regulatory network (GRN) and (b) Inferelator's predicted GRN when applied to ESCAPE data. To aid comparative analysis, when a gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

force exhaustive candidate selection process, the best candidate is determined by evaluating random candidates. The learned tree-based model is used to determine an importance score for each candidate regulator. By using averaging over the trees in the ensemble, the best probability of regulators genes for a target gene is efficiently identified.

GENIE3 (GEne Network Inference with Ensemble of trees) is an algorithm designed to infer a gene regulatory network (GRN) from steady state gene expression data [53]. GENIE3 decomposes the prediction of an n gene GRN into n independent regression problems. The objective of each regression problem is to predict the expression pattern of one unique gene. This gene is treated as a target gene while all other genes are treated as potential regulator genes. Each regression problem uses a tree-based ensemble method, such as Random Forests or Extra-Trees. The significance of a regulator gene in the prediction of the target gene expression pattern determines the likelihood of a regulatory link. Jump3 uses an *on-off* model of gene expression, similar to the Boolean inference method, and estimates the activity state of the promoter of the gene. Jump3 models the expression of a gene using a stochastic differential equation (SDE):

$$dx_i = (A_i\mu_i(t) + b_i - \lambda_i x_i)dt + \sigma dw(t) \tag{3.10}$$

where subscript i represents the ith target gene, $A_i\mu_i(t)$ represents the binary state of the target gene's promoter (on or off), b_i represents the presence of general purpose basal transcription factors, $\lambda_i x_i$ represents decay of the ith target gene and $\sigma dw(t)$ represents a kinetic noise rate.

Jump3 version 2015 was run using MATLAB version 9.1.0.441655 (R2016b) on Windows 8.1 Pro. When Jump3 was applied to DREAM4 data, it predicted a graph containing 43 edges. The top 15 edges were chosen for performance evaluation per

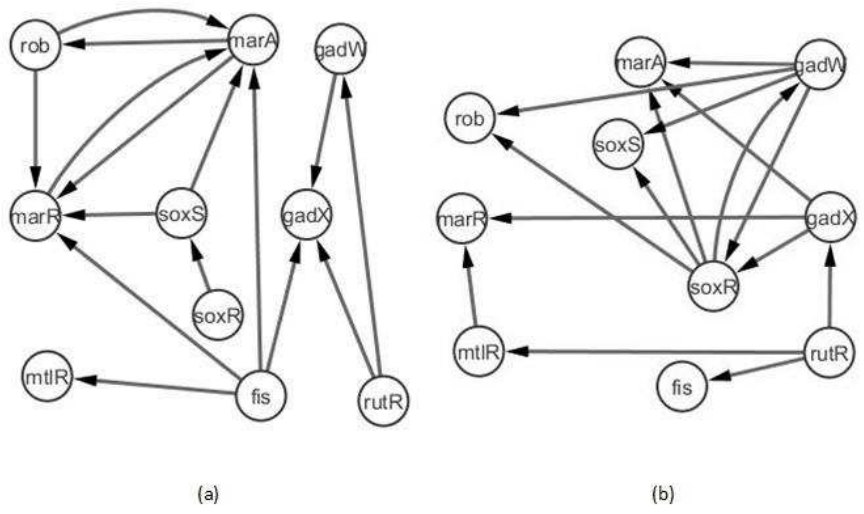


Figure 3.13 Comparison of: (a) DREAM4 gold standard gene regulatory network (GRN) and (b) Jump3's predicted GRN when applied to DREAM4 data. To aid comparative analysis, when a gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

literature recommendation [64]. Jump3 was applied to ESCAPE data and a total of 676 edges were predicted. The top 84 edges were chosen for performance evaluation per the same literature recommendation. Figure 3.13 illustrates how the GRN predicted by Jump3, when applied to DREAM4 data, compares with the DREAM4 gold standard GRN. Similarly, Figure 3.14 illustrates how the GRN predicted by Jump3, when applied to ESCAPE data, compares with the ESCAPE gold standard GRN. For easier comparative analysis of gene interactions in the figures, whenever the same gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

Table 3.1 identifies the seven time-series GRN inference tools used in this survey. The name of each tool, the method used, the tool development platform used, the capability of the tool to apply prior knowledge and a journal citation for the tool are shown.

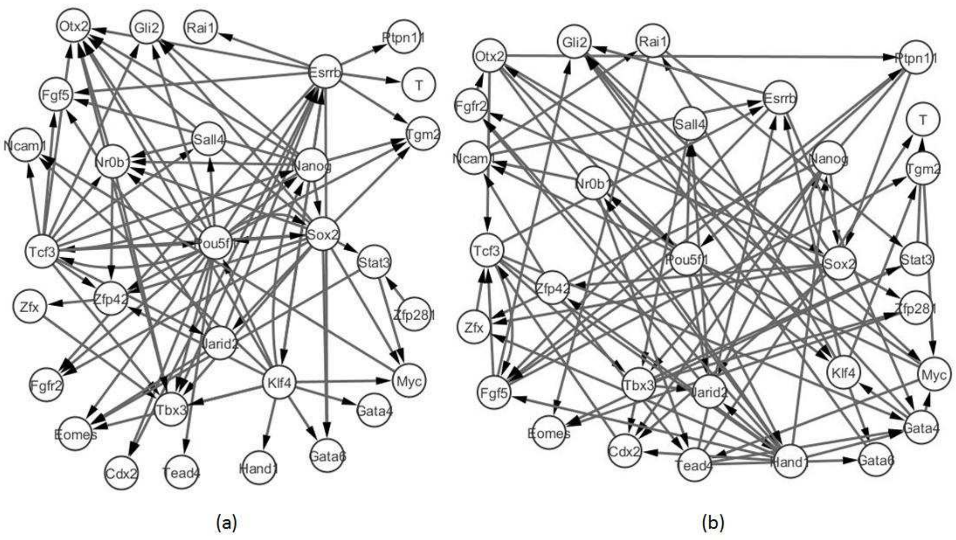


Figure 3.14 Comparison of: (a) ESCAPE gold standard gene regulatory network (GRN) and (b) Jump3's predicted GRN when applied to ESCAPE data. To aid comparative analysis, when a gene appeared in both GRNs, that gene on one GRN was positioned similar to its corresponding gene on the other GRN.

Table 3.1 Time-Series Gene Regulatory Network (GRN) Inference Tools Evaluated

Tool name	**Method***	**Platform**	**Priors?**	**Citation**
MIDER	IT	MATLAB		[111]
TimeDelay-ARACNE	IT	R		[130]
networkBMA	Bayesian	R	Yes	[127]
CGC-2SPR	GC	R/MATLAB	Yes	[124]
BoolNet	Boolean	R	Yes	[80, 100]
Inferelator	ODE	R	Yes	[72]
Jump3	Tree-based	MATLAB	Yes	[53]

*Inference method abbreviations:

GC = Granger causality

IT = Information theoretic

ODE = Ordinary differential equations

3.3 Results

For a fair comparison among the seven GRN inference tools evaluated in this survey, consistent procedures were used whenever possible. Research shows that in spite of the genome complexity diversity among studied organisms, the mean ratio of regulators per gene is between 1.5 and 2 [64]. For this reason, when our testing of a GRN inference tool resulted in regulatory relationships exceeding 15 relationships or 84 relationships for the DREAM4 network or the ESCAPE network, respectively, we chose the top 15 or 84 relationships, respectively. These numbers also match the sizes of the respective DREAM4 and ESCAPE gold standard networks. For each tool surveyed, default runtime parameters were used unless otherwise noted. It's likely that a performance metric score could be improved by fine tuning one or more runtime parameters. However, the intention of the survey was to run each tool "out of the box" to the greatest extent possible. Some of the tools have the capability to use prior knowledge about a network to improve the prediction performance. For a consistent comparison, all tools were run without the benefit of prior network knowledge. Using DREAM4 and ESCAPE gold standard networks as references, performance metrics of the GRN inference tools evaluated in this survey were determined.

In GRN inference, a true positive (TP) indicates that a certain edge is claimed to exist in the gold standard network and the edge is, in fact, present in the gold standard network. A false positive (FP) indicates that a certain edge is claimed to exist in the gold standard network but the edge is, in fact, not present in the gold standard network. A true negative (TN) indicates that a certain edge is claimed to not exist in the gold standard network and the edge is, in fact, not present in the gold standard network. A false negative (FN) indicates that a certain edge is claimed to not exist in the gold standard network but the edge is, in fact, present in the gold standard network. Overall accuracy is defined in Equation (3.11) [122]. We also use TP (FP, TN, FN, respectively) to represent the number of true positives (false

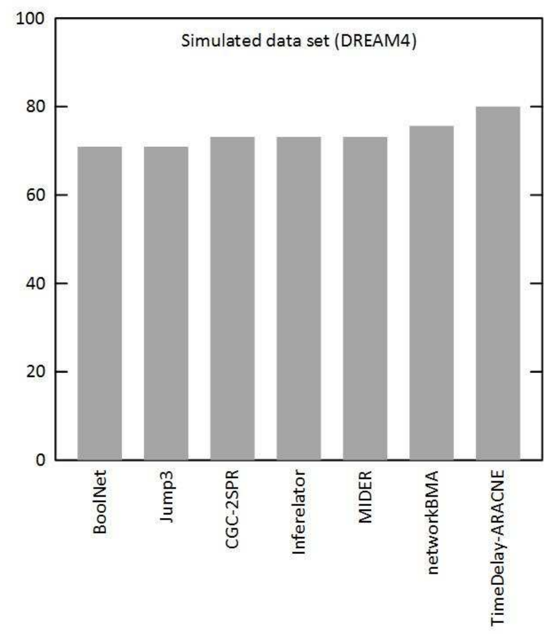


Figure 3.15 Overall accuracy metrics of seven GRN inference tools applied against DREAM4 synthetic time-series gene expression profile.

positives, true negatives, false negatives, respectively), produced by a GRN inference tool. P is the sum of TP and FN. N is the sum of TN and FP.

$$Overall\ accuracy = \frac{TP + TN}{P + N} \tag{3.11}$$

Figure 3.15 shows overall accuracy results for the seven time-series GRN inference tools evaluated in this survey which were applied to the DREAM4 synthetic time-series gene expression profile. The overall accuracy results ranged from 71% for BoolNet and Jump3 to 80% for TimeDelay-ARACNE. Figure 3.16 shows overall accuracy results for the six time-series GRN inference tools evaluated in this survey which were applied to the ESCAPE experimental time-series gene expression profile. The overall accuracy results ranged from 81% for networkBMA to 88% for TimeDelay-ARACNE. Note that no results were produced by the MIDER GRN inference tool when applied to the ESCAPE time-series gene expression profile.

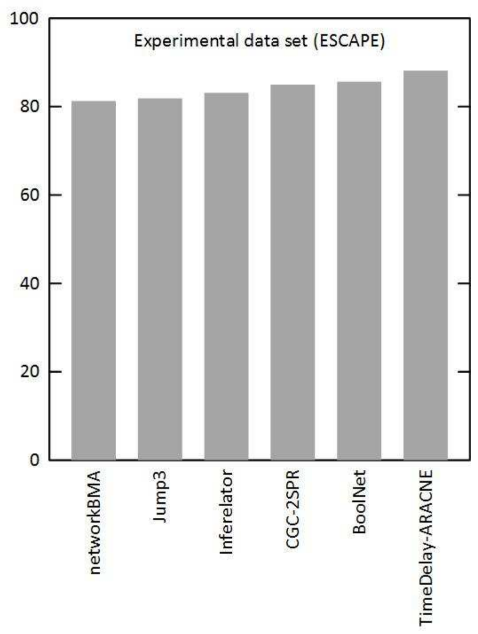


Figure 3.16 Overall accuracy metrics of six GRN inference tools applied against ESCAPE experimental time-series gene expression profile.

Balanced accuracy is defined in Equation (3.12) and is an important metric in a case where a performance estimate may be overly optimistic due to an imbalanced data set [19, 122]. GRNs are sparse graphs in which there are few edges present and many edges absent. This property is known as low connectivity [108]. In the case of *E. Coli*, for example, whose genome is comprised of about 4,000 genes, the mean number of regulatory interactions per gene is 2 to 3. This is a typical imbalanced data set in which the size of the majority class (i.e., the set of edges absent) is much larger than the size of the minority class (i.e., the set of edges present). A blind GRN inference algorithm would work by predicting all edges to be absent. The algorithm would create errors for those few edges present while making correct predictions for all the edges absent, thus still yielding high overall accuracy results. This motivates the need for balanced accuracy, in which the accuracy results for the majority class and minority class are calculated separately.

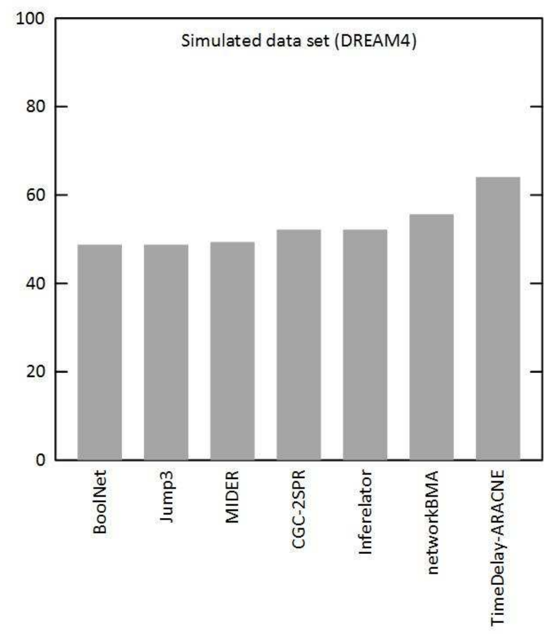


Figure 3.17 Balanced accuracy metrics of seven GRN inference tools applied against DREAM4 synthetic time-series gene expression profile.

$$Balanced\ accuracy = \frac{1}{2}\left(\frac{TP}{P} + \frac{TN}{N}\right) \tag{3.12}$$

Figure 3.17 shows balanced accuracy results for the seven time-series GRN inference tools evaluated in this survey which were applied to the DREAM4 synthetic time-series gene expression profile. The balanced accuracy results ranged from 48% for BoolNet and Jump3 to 64% for TimeDelay-ARACNE using DREAM4 data. Figure 3.18 shows balanced accuracy results for the six time-series GRN inference tools evaluated in this survey which were applied to the ESCAPE experimental time-series gene expression profile. The balanced accuracy results ranged from 47% for networkBMA to 57% for CGC-2SPR using ESCAPE data. Note that no results were produced by the MIDER GRN inference tool when applied to the ESCAPE time-series gene expression profile.

The Precision performance metric (also known as the Positive Predictive Value) is a measure of the relevance of predicted values. If all predicted Positive values were

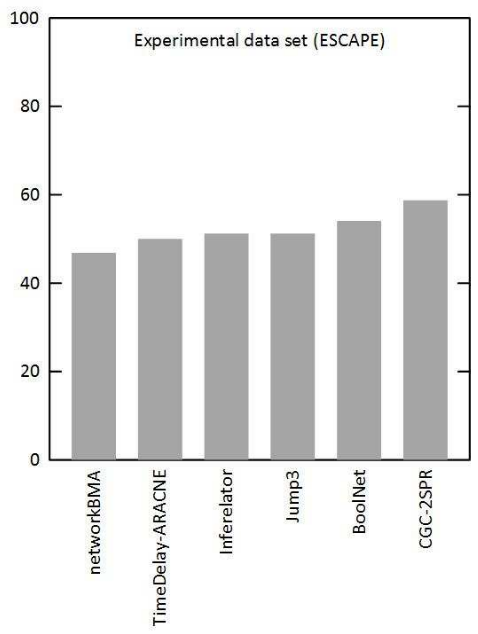


Figure 3.18 Balanced accuracy metrics of six GRN inference tools applied against ESCAPE experimental time-series gene expression profile.

true positives (TP), for instance, then the Precision performance metric would be *100%*. The formula for the Precision performance metric is:

$$Precision = \frac{TP}{TP + FP} \tag{3.13}$$

Precision measures for the seven tools applied to DREAM4 data in this survey ranged from 13% for BoolNet and Jump3 to 40% for TimeDelay-ARACNE with a median Precision value of 20%. Precision measures for the six tools applied to ESCAPE data ranged from 5% for networkBMA to 21% for CGC-2SPR with a median Precision value of 13%.

The Recall performance metric (also known as Sensitivity) is a measure of the relevant values predicted. If all positive values (i.e., gold standard GRN regulatory edges) were predicted, for instance, then the Recall performance metric would be *100%*. The formula for the Recall performance metric is:

$$Recall = \frac{TP}{TP + FN} \tag{3.14}$$

Recall measures for the seven tools applied to DREAM4 data in this survey ranged from 13% for BoolNet, Jump3 and MIDER to 40% for TimeDelay-ARACNE with a median Recall value of 20%. Recall measures for the six tools applied to ESCAPE data ranged from 1% for TimeDelay-ARACNE to 21% for CGC-2SPR with a median Recall value of 12%.

3.4 Discussion

The seven tools used in this survey were chosen to ensure a diverse representation of common time-series GRN inference methods. Each tool was available to be downloaded from a public-access Website. Documented instructions for each tool were sufficient to guide the authors through the necessary steps to run with default parameters and to retrieve results for comparative analysis processing. The balanced accuracy performance metric, as defined in Equation (3.12), addresses the disparity in a typical GRN between the majority and minority classes, i.e., when the data set is imbalanced between edges present and edges absent.

Mining for common patterns is an important research area in many disciplines. For instance, market basket transaction analysis, or affinity analysis, helps determine the likelihood of a shopper buying product y when he or she has already purchased product x [45]. In a similar manner, we propose that among computational biology predictions generated by various GRN inference tools there will be common patterns in the form of frequently occurring gene clusters. We claim that a higher than average presence of a specific cluster of genes is a significant clue to finding the true GRN in the organism being studied.

Motivated by successes using the Apriori algorithm for mining frequent common patterns [4, 128], and by successes combining GRN tools as mentioned in Section 3.1 of this paper, we explored a graph-based frequent pattern mining approach to improve GRN inference performance. We evaluated the seven DREAM4 time-series gene expression data set GRN inference results obtained in this survey (see Section 3.3). DREAM4 GRN inference balanced accuracy results were generally low, with balanced accuracy scores ranging from 48% for BoolNet to 64% for TimeDelay-ARACNE. ESCAPE GRN inference balanced accuracy results were similarly low, with balanced accuracy scores ranging from 47% for networkBMA to 57% for CGC-2SPR. These balanced accuracy performance metrics were based on the gold standard networks obtained for DREAM4 Size10 Network1 [75] and ESCAPE [120], respectively, as described in Section 3.2.

We devised a concept of a common pattern being a network topology among the same genes having at least one edge where the gene count was at least two and the same topology was predicted by a minimum of two GRN inference tools. Our observation was applied to the DREAM4 time-series gene expression data set. One example of this common pattern (using DREAM4 challenge *E. coli* gene names) was the set of two edges $\{marA{\rightarrow}rob, rob{\rightarrow}marA\}$ which were both present, and were the only edges present, in the 2-gene topologies in the GRNs predicted by the Inferelator and networkBMA inference tools. We observed that these two edges were present in the DREAM4 gold standard network.

In a similar manner, we observed common patterns among the results obtained after applying GRN inference tools to the ESCAPE experimental gene expression profile. One example of this common pattern (using ESCAPE database mouse gene names) was the set of two edges $\{Pou5f1{\rightarrow}Nr0b1, Sox2{\rightarrow}Fgfr2\}$ which were both present, and were the only edges present, in the 4-gene topologies in the GRNs predicted by the Inferelator and Jump3 inference tools. We observed that these two edges were present in the ESCAPE gold standard network.

Using a frequent pattern mining approach, results from multiple GRN inference tools can be combined as a means of improving performance metrics. Preliminary but promising results encourage us to pursue this concept of graph-based frequent pattern mining in future GRN research. The combination of multiple GRN inference tools is consistent with recent successful results by others [5, 31, 71, 74, 89].

3.5 Conclusions

We applied seven publicly available and algorithmically disparate GRN inference tools against publicly available time-series gene expression data sets, one experimental and one synthetic. We evaluated the results and observed that scores computed using standard performance metrics were generally low, with, for example, the balanced accuracy scores ranging from a low of 47% for the networkBMA tool applied to the ESCAPE data set to a high of 64% for the TimeDelay-ARACNE tool applied to the DREAM4 data set. We further observed that an improvement in the balanced accuracy performance metric score might be achieved by applying and a graph-based frequent pattern mining approach to the results from all other tools. We conclude that while GRN inference research is intense in a wide variety of interesting approaches, there is no approach or tool that can claim dominating success. We further conclude that group-based efforts have shown, and continue to show, promising advances. Finally, we conclude that a graph-based frequent pattern mining approach, in particular, is promising and is worthy of continued research efforts.

CHAPTER 4

CLOUD-BASED BIOLOGICAL NETWORK INFERENCE: A FRAMEWORK

4.1 Introduction

Unprecedented quantities of biological data generated in a variety of formats at equally unprecedented speeds present a mixture of blessings and challenges to bioinformatics researchers. These three dimensions of challenges, i.e., variety, velocity and volume, comprise the current "big data" phenomenon in bioinformatics [63, 91]. Innovative methodologies in computational biology, including dimensionality reduction, feature selection, parallelization and cloud computing, mitigate high volume complexities by expediting the process of converting big data into useful knowledge [46, 110]. There is no doubt that mining biological big data and revealing details about gene regulatory interactions will lead to a greater understanding of cellular functions.

Time-series gene expression profiles reveal the cell transcriptome at regular time intervals. By mapping the reads in these profiles to their respective reference genome sequence, the source gene for each transcript can be identified. By applying mutual information (MI) methodologies, a directed acyclic graph (DAG) can be constructed to predict a gene regulation network (GRN) for the organism. In the DAG, the two nodes of each edge would represent a pair of genes such that one gene regulates the expression of the second gene. Research in the inference of GRNs produces promising results [76, 130].

No GRN inference research using MI methodologies to date has leveraged the massive parallel processing capability of cloud computing in conjunction with Gene Ontology (GO) knowledge refinement to refine prediction results. We will implement GRN inference using MI methodologies and GO result refinement in a cloud-based environment to predict a gene regulatory network (GRN) from both experimental RNA-Seq and synthetic time-series data sets. Furthermore, we will select data sets for which there are known or predicted GRNs available for benchmarking analysis. We will demonstrate that this cloud-based method can easily be scaled up by multiple orders of magnitude with little impact on performance.

This novel cloud-based implementation to infer a GRN will be an important bioinformatics tool. Cloud resources are increasingly more flexible and affordable compared with local traditional computing resources. Cloud resources are available on demand with minimal financial or time commitment. Cloud computing advantages in the field of bioinformatics research are well known [65].

4.2 Map-Reduce GRN Inference Algorithm Using MI and GO

The open source Apache Hadoop (*http://hadoop.apache.org* - last accessed on 4/2/2017) is used on the Amazon Web Services (AWS) Elastic Cloud Computing (EC2) platform (*http://aws.amazon.com/ec2* - last accessed on 4/3/2017). Hadoop implements Google's MapReduce parallel processing framework [27]. The same process will be followed for synthetic as well as RNA-Seq time series data sets.

The time-series gene expression profiles of each gene represented in a data set will be evaluated to identify a time point at which gene expression changes initially, either positively or negatively, beyond a certain threshold. Then, using an MI approach, a GRN DAG will be constructed depicting the influence that one gene has on another gene. Finally, the DAG will be pruned by consulting the GO database and by applying the Data Processing Inequality (DPI) modifications [130].

As illustrated in Figure 4.1, the Master compute node of the Hadoop implementation will prepare the data for the Slave compute nodes, namely the Map and Reduce functions. For each gene in the gene expression profile, the Master computer node will compare the first gene expression profile value with each subsequent gene expression profile value. When the difference, positive or negative, exceeds a certain threshold percentage, that time point, i.e., the Initial change of Expression (IcE), will be known for that gene [130]. As soon as the expression value exceeds plus or minus the designated threshold percentage of the first gene expression, the IcE value will be known for that gene.

When the IcE is known for all genes, the Master compute node will prepare data partitions for tasks to be performed by Slave compute nodes. In each case where the IcE of a gene is less than or equal to the IcE of a different gene, the Influence (INFL) that the first gene has on the second gene will be determined by a Slave compute node [130]. The computation of INFL for one gene pair is independent of the INFL for a different gene pair. As a result, the computation of INFL for multiple gene pairs will be run simultaneously in parallel. Since the number of these tasks is order n-squared, where n is the number of genes, the power of Hadoop's parallel processing capability will be realized. Using a data set containing 100 genes, the number of gene pairs processed as simultaneous Slave tasks could be as high as 10,000. In theory, a data set containing 20,000 genes generating as many as 400,000,000 Slave tasks should take the same amount of time as a data set containing 10 genes, assuming that sufficient cloud resources are available for the simultaneous Slave tasks.

As prescribed by the MapReduce framework, a Map function in Hadoop is defined to process data in the form of a pair of data items. For this research, the first of the two items comprising the Map input data pair will be a unique gene-pair identifier, such that the first gene has an IcE value which is less than the IcE value of the second gene in the gene-pair. The gene-ID of the first gene in the gene-pair will be the same for all gene-pairs assigned to one Map task. The second of the two items comprising the Map input data pair will be the two complete sets of gene expressions for all time points for both genes in the gene-pair.

As prescribed by the MapReduce framework, a Map function in Hadoop is defined to produce data for the Reduce function in the form of another pair of data items. For this research, the first of the two data items comprising the Map output data pair will be the gene-ID of the first gene in the gene-pair that was processed by a Map function. This ensures that each gene under consideration as a regulator of other genes will be processed by only one Reduce task. The second of the two items comprising the Map output data pair will be second gene in the gene-pair and the INFL computed for the gene-pair, i.e., a numeric expression of the influence that the first gene has on the second gene. Simultaneously, the INFL value will be evaluated. All gene-pairs having an INFL value below a certain threshold (to be determined), will be discarded and not forward to a Reduce function for further processing. Thus, if a gene is not deemed to have a sufficient influence on another gene, there will be no edge for that gene-pair on the inferred gene regulatory network.

As prescribed by the MapReduce framework, a Reduce function in Hadoop is defined to process data in the form of a pair of data items produced by the Map function described above. For this research, the Reduce function will consolidate all Mapper-produced gene-pairs according to the first gene in each gene-pair. Furthermore, the Reduce task will evaluate the GO database and determine of the first gene in the gene-pair has been shown to possess a regulatory capability. This information will be valuable in "certifying" that a DAG edge in the GRN should be considered was high confidence. The Reducer nodes will produce one output key-value pair for each unique "first-gene" among all gene-pairs. Thus, the Master node will receive the same number of key-value data pairs as there are unique "first-genes" among all gene-pairs.

The key-value data pair produced by the Reducer nodes is described as follows. The "key" in each key-value data pair will be a unique numeric gene-id. This gene-id will represent all gene-pairs that have the same gene-id as the first of the two genes

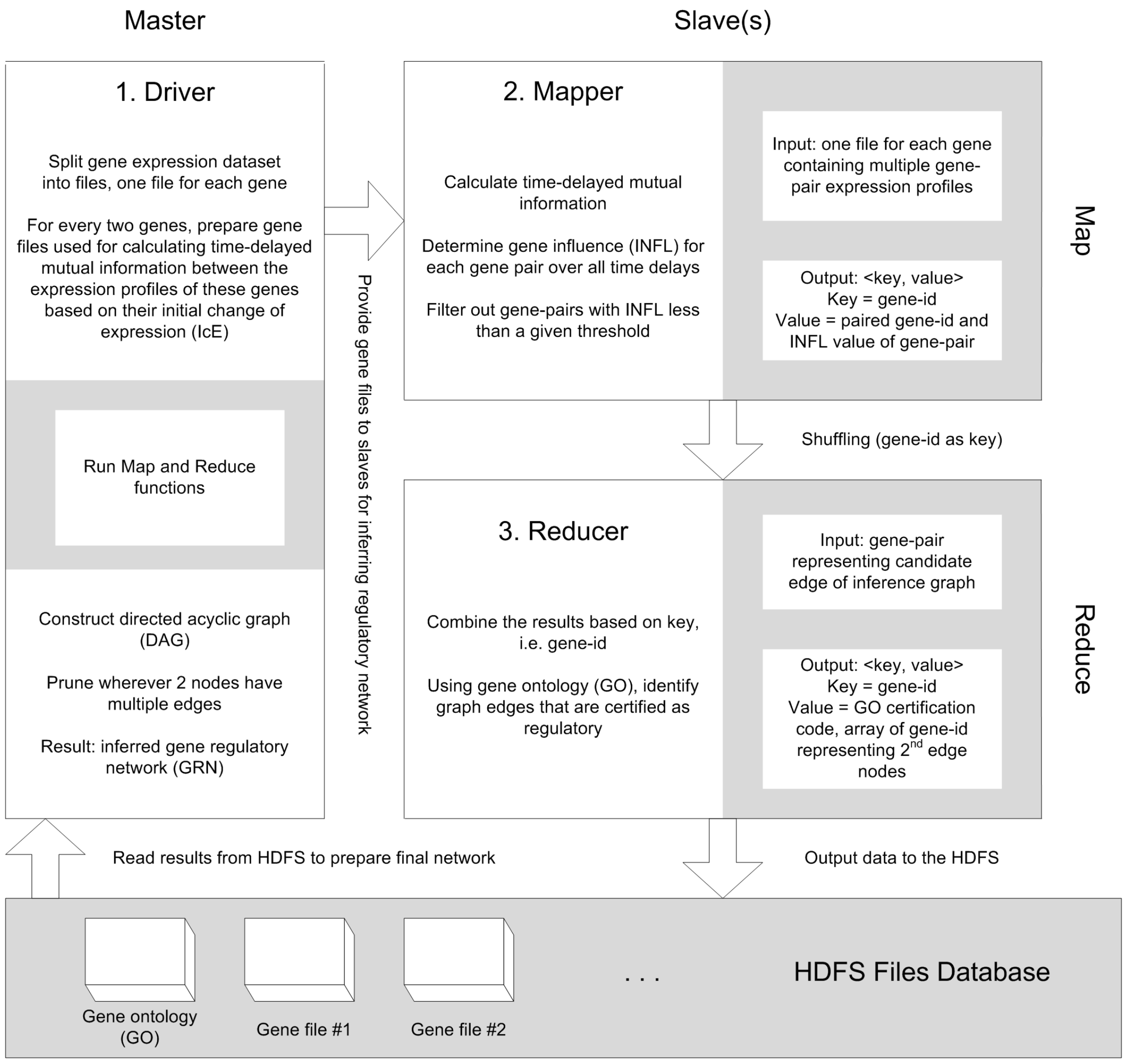

Master
Slave(s)
1. Driver
Split gene expression dataset into files, one file for each gene
For every two genes, prepare gene files used for calculating time-delayed mutual information between the expression profiles of these genes based on their initial change of expression (IcE)
Run Map and Reduce functions
Construct directed acyclic graph (DAG)
Prune wherever 2 nodes have multiple edges
Result: inferred gene regulatory network (GRN)
Provide gene files to slaves for inferring regulatory network
2. Mapper
Calculate time-delayed mutual information
Determine gene influence (INFL) for each gene pair over all time delays
Filter out gene-pairs with INFL less than a given threshold
Input: one file for each gene containing multiple gene-pair expression profiles
Output: <key, value> Key = gene-id Value = paired gene-id and INFL value of gene-pair
Map
Shuffling (gene-id as key)
3. Reducer
Combine the results based on key, i.e. gene-id
Using gene ontology (GO), identify graph edges that are certified as regulatory
Input: gene-pair representing candidate edge of inference graph
Output: <key, value> Key = gene-id Value = GO certification code, array of gene-id representing 2nd edge nodes
Reduce
Read results from HDFS to prepare final network
Output data to the HDFS
Gene ontology (GO)
Gene file #1
Gene file #2
. . .
HDFS Files Database

Figure 4.1 MapReduce framework showing network inference with Gene Ontology (GO) edge certification.

in the gene-pairs. The “value” in each key-value data pair will be an array of data triples. The first item in each data triple in this array will be the second gene of the gene-pair produced by the Mapper nodes. The second item in each data triple in this array will be the INFL value associated with each gene-pair produced by the Mapper nodes. The third item in each data triple in this array will be the GO certification value associated with each gene-pair produced by the Mapper nodes. GO certification will be a simple binary value, i.e., YES or NO. The output produced by the Reduce function will be evaluated by the Master compute node.

At the conclusion of all Map and Reduce tasks, the Master compute node will examine the collection of gene-pair INFL and GO certification values. This information will be used to create a DAG comprised of a directed edge for each gene-pair in the collection. If the GO database confirms that the gene has a regulatory function, then the edge in the DAG will be marked as a “GO certified” edge. The DAG will be checked for pairs of genes that have a directed edge in both directions. In each such case, one edge will be removed if it is not a “GO certified” edge and if the INFL of the edge is less than the INFL of the second edge.

The resultant DAG will represent a prediction of a GRN for the organism represented by the gene expression profile. This prediction will be benchmarked against similar predictions compiled for the same gene expression profile.

We have received some results using the proposed cloud-based framework [3].

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

Research shows that when prior knowledge is available and applied in gene regulatory network inference (GRN), network prediction performance results are generally improved [21]. Seven recent bioinformatics tools for time-series based GRN inference were reviewed and compared using both simulated and experimental data sets. Standard performance metrics for these GRN inference tools are generally low, suggesting that further efforts be needed to develop more reliable network inference tools. Using multiple tools together can help identify true regulatory interactions between genes, a finding consistent with those reported in the literature.

In the future, we plan to explore and evaluate new algorithms leveraging frequent subgraph mining (FSM) for genome-wide pattern discovery and network inference using steady-state and time series data in the cloud. We will also perform network inference algorithms using Apache Spark and Apache Hadoop and compare them [27, 98].

REFERENCES

[1] D. P. Aalberts and N. O. Hodas. Asymmetry in RNA pseudoknots: observation and theory. *Nucleic Acids Research*, 33:2210–2214, 2005.

[2] D. P. Aalberts and N. Nandagopal. A two-length-scale polymer theory for RNA loop free energies and helix stacking. *RNA*, 16:1350–1355, 2010.

[3] Y. Abduallah, T. Turki, K. Byron, Z. Du, M. Cervantes-Cervantes, and J. T. L. Wang. MapReduce algorithms for inferring gene regulatory networks from time-series microarray data using an information-theoretic approach. *BioMed Research International*, 2017.

[4] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proceedings 20th international conference very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[5] G. Altay and F. Emmert-Streib. Revealing differences in gene network inference algorithms on the network level by ensemble methods. *Bioinformatics*, 26(14):1738–1744, 2010.

[6] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[7] G. M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of ACM Spring Joint Computer Conference*, pages 483–485, New York, 1967.

[8] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock. Gene Ontology: tool for the unification of biology. *Nature Genetics*, 25(1):25–29, 2000.

[9] N. Ban, P. Nissen, J. Hansen, P. B. Moore, and T. A. Steitz. The complete atomic structure of the large ribosomal subunit at 2.4 A resolution. *Science*, 289:905–920, 2000.

[10] S. Barman and Y. K. Kwon. A novel mutual information-based Boolean network inference method from time-series gene expression data. *PLoS ONE*, 12(2), 2017.

[11] D. A. Benson, M. Cavanaugh, K. Clark, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and E. W. Sayers. GenBank. *Nucleic Acids Research*, 41:D36–D42, 2013.

[12] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic Acids Research*, 28:235–242, 2000.

[13] J. A. Blake, M. Dolan, H. Drabkin, D. P. Hill, L. Ni, D. Sitnikov, S. Burgess, T. Buza, C. Gresham, F. McCarthy, and others. Gene Ontology: enhancements for 2011. *Nucleic Acids Research*, 40(D1):D559–D564, 2012.

[14] J. A. Blake and M. A. Harris. Gene Ontology (GO) project: structured vocabularies for molecular biology and their application to genome and expression analysis. *Current Protocols in Bioinformatics*, Chapter 7, 2008.

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

[16] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[17] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Belmont, CA: Wadsworth, 1984.

[18] S. L. Bressler and A. K. Seth. Wiener-Granger causality: a well established methodology. *Neuroimage*, 58(2):323–329, 2011.

[19] K. H. Brodersen, C. S. Ong, J. M. Buhmann, and K. E. Stephan. The balanced accuracy and its posterior distribution. In *Proceedings - International Conference on Pattern Recognition*, pages 3121–3124, Department of Computer Science, ETH Zurich, 2010.

[20] K. Byron, M. Cervantes-Cervantes, J. T. L. Wang, W. C. Lin, and Y. Park. Mining *roX1* RNA in *Drosophila* genomes using covariance models. *International Journal of Computational Bioscience*, 1(1):22–32, 2010.

[21] K. Byron, K. G. Herbert, and J. T. L. Wang. *Bioinformatics Database Systems*. Abingdon, UK: Taylor & Francis, 2017.

[22] V. Chandrasekaran, N. Srebro, and P. Harsha. Complexity of inference in graphical models. In *Proceedings of 24th Conference on Uncertainty in Artificial Intelligence*, pages 70–78, Massachusetts Institute of Technology, 2008.

[23] FlyBase Consortium. FlyBase: the *Drosophila* database. *Nucleic Acids Research*, 22(17):3456–3458, 1994.

[24] G. F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.

[25] F. F. Costa. Non-coding RNAs: lost in translation? *Gene*, 386(1-2):1–10, 2007.

[26] K. Darty, A. Denise, and Y. Ponty. VARNA: interactive drawing and editing of the RNA secondary structure. *Bioinformatics*, 25(15):1974–1975, 2009.

[27] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. *Communications of ACM*, 51(1):107–113, 2008.

[28] R. Durbin, S. R. Eddy, A. Krogh, and G. J. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge, UK: Cambridge University Press, 1998.

[29] S. R. Eddy. Non-coding RNA genes and the modern RNA world. *Nature Reviews. Genetics*, 2(12):919–929, 2001.

[30] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of National Academy of Sciences of the United States of America*, (25), 1998.

[31] R. J. Flassig, S. Heise, K. Sundmacher, and S. Klamt. An effective framework for reconstructing gene regulatory networks from genetical genomics data. *Bioinformatics*, 29(2):246–254, 2013.

[32] E. K. Freyhult, J. P. Bollback, and P. P. Gardner. Exploring genomic dark matter: A critical assessment of the performance of homology search methods on noncoding RNA. *Genome Research*, 17(1):117–125, 2007.

[33] N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3-4):601–620, 2000.

[34] A. Fujita, J. R. Sato, H. M. Garay-Malpartida, P. A. Morettin, M. C. Sogayar, and C. E. Ferreira. Time-varying modeling of gene expression regulatory networks using the wavelet dynamic vector autoregressive method. *Bioinformatics*, 23(13):1623–1630, 2007.

[35] T. S. Gardner, D. di Bernardo, D. Lorenz, and J. J. Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301(5629):102–105, 2003.

[36] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[37] S. Ghemawat, H. Gobioff, and S. Leung. The Google file system. *SIGOPS Operating Systems Review*, 37(5):29–43, 2003.

[38] J. Goecks, J. Taylor, A. Nekrutenko, E. Afgan, G. Ananda, D. Baker, D. Blankenberg, R. Chakrabarty, N. Coraor, J. Goecks, G. Von Kuster, R. Lazarus, K. Li, A. Nekrutenko, J. Taylor, and K. Vincent. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biology*, 11(8), 2010.

[39] C. W. J. Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, (3), 1969.

[40] A. Greenfield, R. Bonneau, A. Madar, and H. Ostrer. DREAM4: Combining genetic and dynamic information to identify biological networks and dynamical models. *PLoS One*, 5(10), 2010.

[41] S. Griffiths-Jones, A. Bateman, M. Marshall, A. Khanna, and S. R. Eddy. Rfam: an RNA family database. *Nucleic Acids Research*, 31:439–441, 2003.

[42] S. Griffiths-Jones, A. Khanna, S. R. Eddy, S. Moxon, M. Marshall, and A. Bateman. Rfam: Annotating non-coding RNAs in complete genomes. *Nucleic Acids Research*, 33:D121–D124, 2005.

[43] T. Gunarathne, B. Zhang, T. Wu, and J. Qiu. Scalable parallel computing on clouds using Twister4Azure iterative MapReduce. *Future Generation Computer Systems*, 29(4):1035–1048, 2013.

[44] J. D. Hamilton. *Time Series Analysis.* Princeton, NJ: Princeton University Press, 1994.

[45] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques. 3rd ed.* San Francisco, CA: Morgan Kaufmann, 2011.

[46] Y. Han, K. Muegge, S. Gao, W. Zhang, and B. Zhou. Advanced applications of RNA sequencing and challenges. *Bioinformatics and Biology Insights*, 9:29–46, 2015.

[47] D. J. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining.* Cambridge, MA: MIT Press, 2001.

[48] A. Herraez. Biomolecules in the computer: Jmol to the rescue. *Biochemical Education*, 34:255–261, 2006.

[49] M. Höchsmann, T. Töller, R. Giegerich, and S. Kurtz. Local similarity in RNA secondary structures. In *Proceedings of IEEE Computer Society Bioinformatics Conference*, volume 2, pages 159–168, 2003.

[50] I. L. Hofacker. Vienna RNA secondary structure server. *Nucleic Acids Research*, 31(13):3429–3431, 2003.

[51] C. Huang and A. Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):255–263, 1996.

[52] V. A. Huynh-Thu, A. Irrthum, L. Wehenkel, and P. Geurts. Inferring regulatory networks from expression data using tree-based methods. *PLoS One*, 5(9):1–10, 09 2010.

[53] V. A. Huynh-Thu and G. Sanguinetti. Combining tree-based and dynamical systems for the inference of gene regulatory networks. *Bioinformatics*, 31(10):1614–1622, 2014.

[54] F. V. Jensen. *An Introduction to Bayesian Networks.* New York, NY: Springer, 1996.

[55] L. A. Kavanaugh and F. S. Dietrich. Non-coding RNA prediction and verification in *Saccharomyces cerevisiae*. *PLoS Genetics*, 5(1), 2009.

[56] M. Khaladkar, J. Liu, D. Wen, J. T. L. Wang, and B. Tian. Mining small RNA structure elements in untranslated regions of human and mouse mRNAs using structure-based alignment. *BMC Genomics*, 9, 2008.

[57] M. Khaladkar, V. Patel, V. Bellofatto, J. Wilusz, and J. T. L. Wang. Detecting conserved secondary structures in RNA molecules using constrained structural alignment. *Computational Biology and Chemistry*, 32(4):264–272, 2008.

[58] M. Khaladkar, J. T. L. Wang, V. Bellofatto, B. Tian, and B. A. Shapiro. RADAR: A Web server for RNA data analysis and research. *Nucleic Acids Research*, 35:W300–W304, 2007.

[59] J. Kim, A. E. Walter, and D. H. Turner. Thermodynamics of coaxially stacked helixes with GA and CC mismatches. *Biochemistry*, 35:13753–13761, 1996.

[60] S. H. Kim, J. L. Sussman, F. L. Suddath, G. J. Quigley, A. McPherson, A. H. J. Wang, N. C. Seeman, and A. Rich. The general structure of transfer RNA molecules. *Proceedings of National Academy of Sciences of the United States of America*, (12):4970–4974, 1974.

[61] S. Y. Kim, S. Imoto, and S. Miyano. Inferring gene networks from time series microarray data using dynamic Bayesian networks. *Briefings in bioinformatics*, 4(3):228–235, 2003.

[62] C. Laing, D. Wen, J. T. L. Wang, and T. Schlick. Predicting coaxial helical stacking in RNA junctions. *Nucleic Acids Research*, 40:487–498, 2012.

[63] D. Laney. 3D data management: Controlling data volume, velocity and variety. *META Group Research Note*, 6, 2001.

[64] R. D. Leclerc. Survival of the sparsest: Robust gene networks are parsimonious. *Molecular Systems Biology*, 4, 2008.

[65] Y. Lee, Y. Hsiao, and W. Hwang. Designing a parallel evolutionary algorithm for inferring gene networks on the cloud computing environment. *BMC Systems Biology*, 8, 2014.

[66] N. B. Leontis, A. Lescoute, and E. Westhof. The building blocks and motifs of RNA architecture. *Current Opinions in Structural Biology*, 16:279–287, 2006.

[67] A. Lescoute and E. Westhof. Topology of three-way junctions in folded RNAs. *RNA*, 12(1):83–93, 2006.

[68] S. Liang, S. Fuhrman, and R. Somogyi. Reveal, a general reverse engineering algorithm for inference of genetic network architectures. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 18–29, 1998.

[69] J. M. Lingeman and D. Shasha. *Network Inference in Molecular Biology: A Hands-on Framework*. New York, NY: Springer, 2012.

[70] J. Liu, J. T. L. Wang, J. Hu, and B. Tian. A method for aligning RNA secondary structures and its application to RNA motif detection. *BMC Bioinformatics*, 6, 2005.

[71] A. Madar, R. Bonneau, A. Greenfield, and E. Vanden-Eijnden. DREAM3: network inference using dynamic context likelihood of relatedness and the inferelator. *PLoS ONE*, 5(3), 2010.

[72] A. Madar, A. Greenfield, H. Ostrer, E. Vanden-Eijnden, and R. Bonneau. The Inferelator 2.0: a scalable framework for reconstruction of dynamic regulatory network models. *Annual International Conference of the IEEE Engineering in Medicine and Biology Society.*, pages 5448–5451, 2009.

[73] D. Marbach, R. Küffner, M. Kellis, B. Holmes, J. C. Costello, N. M. Vega, D. M. Camacho, K. R. Allison, J. J. Collins, N. Vega, T. Petri, L. Windhager, R. Zimmer, R. J. Prill, G. Stolovitzky, A. Aderhold, R. Bonneau, F. Dondelinger, D. Husmeier, A. Madar, C. S. Poultney, A. Greenfield, S. Mani, Y. Chen, F. Cordero, R. Esposito, A. Visconti, M. Crane, H. J. Ruskin, A. Sîrbu, M. Drton, R. Foygel, S. Rezny, A. De La Fuente, V. De Leo, A. Pinna, N. Soranzo, J. Gertheiss, T. Hothorn, P. Geurts, V. A. Huynh-Thu, A. Irrthum, L. Wehenkel, M. Grzegorczyk, A. C. Haury, F. Mordelet, P. Vera-Licona, J. P. Vert, G. Karlebach, R. Shamir, S. Lèbre, H. Ostrer, Z. Ouyang, M. Song, H. Wang, Y. Zhang, R. Pandya, Y. Saeys, and A. Statnikov. Wisdom of crowds for robust gene network inference. *Nature Methods*, 9(8):796–804, 2012.

[74] D Marbach, S Roy, F Ay, PE Meyer, R Candeias, T Kahveci, CA Bristow, and M Kellis. Predictive regulatory models in *Drosophila melanogaster* by integrative inference of transcriptional networks. *Genome Research*, 22:1334–1349, 2012.

[75] D. Marbach, T. Schaffter, C. Mattiussi, and D. Floreano. Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *Journal of Computational Biology*, 16(2):229–239, 2009.

[76] A. A. Margolin, A. Califano, I. Nemenman, C. Wiggins, K. Basso, R. D. Favera, and G. Stolovitzky. ARACNE: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC Bioinformatics*, 7, 2006.

[77] J. S. Mattick and I. V. Makunin. Non-coding RNA. *Human Molecular Genetics*, 15:R17–R29, 2006.

[78] N. D. Mukhopadhyay and S. Chatterjee. Causality and pathway search in microarray time series experiment. *Bioinformatics*, 23(4):442–449, 2007.

[79] K. Murphy and S. Mian. Modelling gene expression data using dynamic Bayesian networks. Technical report, Computer Science Division, University of California, Berkeley, CA, 1999.

[80] C. Müssel, H. A. Kestler, and M. Hopfensitz. BoolNet-an R package for generation, reconstruction and analysis of Boolean networks. *Bioinformatics*, 26(10):1378–1380, 2010.

[81] R. Nagarajan and M. Upreti. Comment on causality and pathway search in microarray time series experiment. *Bioinformatics*, 24(7):1029–1032, 2008.

[82] E. P. Nawrocki and S. R. Eddy. Infernal 1.1: 100-fold faster RNA homology searches. *Bioinformatics*, 29(22):2933–2935, 2013.

[83] S. W. Park, Y. Park, and M. I. Kuroda. Regulation of histone H4 Lys16 acetylation by predicted alternative secondary structures in *roX* noncoding RNAs. *Molecular and Cellular Biology*, 28(16):4952–4962, 2008.

[84] S. W. Park, I. K. Yool, J. G. Sypula, J. Choi, H. Oh, and Y. Park. An evolutionarily conserved domain of *roX2* RNA is sufficient for induction of H4-Lys16 acetylation on the *Drosophila* X chromosome. *Genetics*, 177(3):1429–1437, 2007.

[85] Y. Park, R. L. Kelley, H. Oh, M. I. Kuroda, and V. H. Meller. Extent of chromatin spreading determined by *roX* RNA recruitment of MSL proteins. *Science*, (5598):1620–1623, 2002.

[86] Y. Park and M. I. Kuroda. Epigenetic aspects of X-chromosome dosage compensation. *Science*, (5532), 2001.

[87] N. Patel and J. T. L. Wang. Semi-supervised prediction of gene regulatory networks using machine learning algorithms. *Journal of Biosciences*, 40(4):731–740, 2015.

[88] L. Peng, L. K. Ng, and S. See. YellowRiver: A flexible high performance cluster computing service for grid. In *Proceedings - Eighth International Conference on High-Performance Computing in Asia-Pacific Region*, volume 2005, pages 553–558, Nanyang, 2005.

[89] R. J. Prill, J. Saez-Rodriguez, L. G. Alexopoulos, P. K. Sorger, and G. Stolovitzky. Crowdsourcing network inference: the DREAM predictive signaling network challenge. *Science Signaling*, 4(189), 2011.

[90] A. M. Pyle and Z. Shakked. The ever-growing complexity of nucleic acids: from small DNA and RNA motifs to large molecular assemblies and machines (editorial overview). *Current Opinions in Structural Biology*, 21:293–295, 2011.

[91] Y. Qin, H. K. Yalamanchili, J. Qin, B. Yan, and J. Wang. The current status and challenges in computational analysis of genomic big data. *Big Data Research*, 2:12–18, 2015.

[92] N. J. Reiter, C. W. Chan, and A. Mondrago. Emerging structural themes in large RNA molecules. *Current Opinions in Structural Biology*, 21:319–326, 2011.

[93] J. S. Reuter and D. H. Mathews. RNAstructure: software for RNA secondary structure prediction and analysis. *BMC Bioinformatics*, 11, 2010.

[94] M. A. Savageau. Introduction to S-systems and the underlying power-law formalism. *Mathematical and Computer Modelling*, 11(C):546–551, 1988.

[95] S. M. Savaresi, D. L. Boley, S. Bittanti, and G. Gazzaniga. Cluster selection in divisive clustering algorithms. In *SIAM International Conference on Data Mining*, pages 299–314, 2002.

[96] F. Schluenzen, A. Tocilj, J. Harms, M. Gluehmann, D. Janell, A. Yonath, R. Zarivach, A. Bashan, H. Bartels, I. Agmon, and F. Franceschi. Structure of functionally activated small ribosomal subunit at 3.3 Å resolution. *Cell*, 102(5):615–623, 2000.

[97] I. Shmulevich, W. Zhang, E. R. Dougherty, and S. Kim. Probabilistic Boolean networks: A rule-based uncertainty model for gene regulatory networks. *Bioinformatics*, 18(2):261–274, 2002.

[98] A. G. Shoro and Tariq R. Soomro. Big data analysis: Apache Spark perspective. *Global Journal of Computer Science and Technology*, 15(1), 2015.

[99] A. C. G. Silveira, F. L. Thompson, A. T. R. Vasconcelos, K. L. Robertson, B. Lin, Z. Wang, and G. J. Vora. Identification of non-coding RNAs in environmental *Vibrios*. *Microbiology*, 156(8):2452–2458, 2010.

[100] A. Singh, J. M. Nascimento, S. Kowar, H. Busch, and M. Boerries. Boolean approach to signalling pathway modelling in HGF-induced keratinocyte migration. *Bioinformatics*, 28(18):i495–i501, 2012.

[101] A. Stark, M. F. Lin, M. Kellis, P. Kheradpour, M. D. Rasmussen, A. N. Deoras, J. S. Pedersen, A. S. Hinrichs, B. Paten, W. J. Kent, D. Haussler, L. Parts, J. W. Carlson, S. E. Celniker, C. Yu, S. Park, K. H. Wan, M. A. Crosby, W. M. Gelbart, B. B. Matthews, A. J. Schroeder, L. S. Gramates, S. E. St Pierre, M. Roark, K. L. Wiley Jr., R. J. Kulathinal, P. Zhang, K. V. Myrick, J. V. Antone, S. Roy, J. G. Ruby, D. P. Bartel, J. Brennecke, E. Hodges, G. J. Hannon, A. Caspi, S. W. Park, Y. Park, M. V. Han, M. W. Hahn, M. L. Maeder, B. J. Polansky, B. E. Robson, D. A. Eastman, S. Aerts, B. Hassan, J. Van Helden, D. G. Gilbert, T. C. Kaufman, M. Rice, M. Weir, C. N. Dewey, L. Pachter, E. C. Lai, M. B. Eisen, A. G. Clark, and D. Smith. Discovery of functional elements in 12 *Drosophila* genomes using evolutionary signatures. *Nature*, 450(7167):219–232, 2007.

[102] G. Storz. An expanding universe of noncoding RNAs. *Science*, (5571), 2002.

[103] C. Stuckenholz, M. I. Kuroda, and V. H. Meller. Functional redundancy within *roX1*, a noncoding RNA involved in dosage compensation in *Drosophila melanogaster*. *Genetics*, 164(3):1003–1014, 2003.

[104] A. R. Subramanian, M. Kaufmann, and B. Morgenstern. DIALIGN-TX: Greedy and progressive approaches for segment-based multiple sequence alignment. *Algorithms for Molecular Biology*, 3(1), 2008.

[105] Y. Sun, J. Buhler, and C. Yuan. Designing filters for fast-known ncRNA identification. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 9(3):774–787, 2012.

[106] Y. Tabei, K. Asai, H. Kiryu, and T. Kin. A fast structural multiple alignment method for long RNA sequences. *BMC Bioinformatics*, 9, 2008.

[107] P. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining.* Boston, MA: Addison-Wesley, 2006.

[108] D. Thieffry, A. M. Huerta, E. Pérez-Rueda, and J. Collado-Vides. From specific gene regulation to genomic networks: a global analysis of transcriptional regulation in *Escherichia coli. Bioessays: News And Reviews In Molecular, Cellular And Developmental Biology*, 20(5):433–440, 1998.

[109] N. Toor, K. S. Keating, S. D. Taylor, and A. M. Pyle. Crystal structure of a self-spliced group II intron. *Science*, 320:77–82, 2008.

[110] I. Triguero, S. del Río, V. López, J. Bacardit, J. M. Benítez, and F. Herrera. ROSEFW-RF: the winner algorithm for the ECBDL'14 big data competition: An extremely imbalanced big data bioinformatics problem. *Knowledge-Based Systems*, 87:69–79, 2015.

[111] A. F. Villaverde, J. R. Banga, J. Ross, and F. Morán. MIDER: Network inference with mutual information distance and entropy reduction. *PLoS ONE*, 9(5), 2014.

[112] A. E. Walter, D. H. Turner, J. Kim, M. H. Lyttle, P. Müller, D. H. Mathews, and M. Zuker. Coaxial stacking of helixes enhances binding of oligoribonucleotides and improves predictions of RNA folding. *Proceedings of National Academy of Sciences of the United States of America*, 91(20):9218–9222, 1994.

[113] A. X. Wang, W. L. Ruzzo, and M. Tompa. How accurately is ncRNA aligned within whole-genome multiple alignments?. *BMC Bioinformatics*, 8, 2007.

[114] Z. Weinberg and W. L. Ruzzo. Sequence-based heuristics for faster annotation of non-coding RNA families. *Bioinformatics*, 22(1):35–39, 2006.

[115] B. T. Wimberly, D. E. Brodersen, W. M. Clemons Jr., R. J. Morgan-Warren, A. P. Carter, V. Ramakrishnan, C. Vonrheln, and T. Hartsch. Structure of the 30S ribosomal subunit. *Nature*, 407(6802):327–339, 2000.

[116] T. K. Wong, T. W. Lam, W. K. Sung, and S. M. Yiu. Adjacent nucleotide dependence in ncRNA and order-1 SCFG for ncRNA identification. *PLoS One*, 5(9), 2010.

[117] Y. Xiao. A tutorial on analysis and simulation of Boolean gene regulatory network models. *Current Genomics*, 10(7):511–525, 2009.

[118] Y. Xiao and E. R. Dougherty. Optimizing consistency-based design of context-sensitive gene regulatory networks. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 53(11):2431–2437, 2006.

[119] Y. Xin, C. Laing, N. B. Leontis, and T. Schlick. Annotation of tertiary interactions in RNA structures reveals variations and correlations. *RNA*, 14:2465–2477, 2008.

[120] H. Xu, C. Baroukh, R. Dannenfelser, E. Y. Chen, C. M. Tan, Y. Kou, Y. E. Kim, I. R. Lemischka, and A. Ma'ayan. ESCAPE: database for integrating high-content published data collected from human and mouse embryonic stem cells. *Database: The Journal of Biological Databases & Curation*, 2013:1–12, 2013.

[121] H. Xu, I. R. Lemischka, A. Ma'ayan, Y. S. Ang, and A. Sevilla. Construction and validation of a regulatory network for pluripotency and self-renewal of mouse embryonic stem cells. *PLoS Computational Biology*, 10(8), 2014.

[122] S. Xu, C. Markson, K. L. Costello, C. Y. Xing, K. Demissie, and A. A. Llanos. Leveraging social media to promote public health knowledge: Example of cancer awareness via Twitter. *JMIR Public Health And Surveillance*, 2(1), 2016.

[123] H. Yang, F. Jossinet, N. Leontis, L. Chen, J. Westbrook, H. Berman, and E. Westhof. Tools for the automatic identification and classification of RNA base pairs. *Nucleic Acids Research*, 31(13):3450–3460, 2003.

[124] S. Yao, S. Yoo, and D. Yu. Prior knowledge driven Granger causality analysis on gene regulatory network discovery. *BMC Bioinformatics*, 16(1), 2015.

[125] Z. Yao, Z. Weinberg, and W.L. Ruzzo. CMfinder: a covariance model based RNA motif finding algorithm. *Bioinformatics*, 22(4):445–452, 2006.

[126] K. Y. Yeung, R. E. Bumgarner, and A. E. Raftery. Bayesian model averaging: Development of an improved multi-class, gene selection and classification tool for microarray data. *Bioinformatics*, 21(10):2394–2402, 2005.

[127] W. C. Young, A. E. Raftery, and K. Y. Yeung. Fast Bayesian inference for gene regulatory networks using ScanBMA. *BMC Systems Biology*, 8(1), 2014.

[128] S. Zhang, Z. Du, and J. T. L. Wang. New techniques for mining frequent patterns in unordered trees. *IEEE Transactions on Cybernetics*, 45(6):1113–1125, 2015.

[129] S. Zhao, K. Prenger, and L. Smith. Stormbow: A cloud-based tool for read mapping and expression quantification in large-scale RNA-seq studies. *ISRN Bioinformatics*, 2013.

[130] P. Zoppoli, S. Morganella, and M. Ceccarelli. TimeDelay-ARACNE: Reverse engineering of gene networks from time-course data by an information theoretic approach. *BMC Bioinformatics*, 11, 2010.

[131] C. Zou and J. Feng. Granger causality vs. dynamic Bayesian network inference: A comparative study. *BMC Bioinformatics*, 10, 2009.