Improving Reconstruction Efficiency of Time-varying Gene Regulatory Networks

Thesis submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

by

Saptarshi Pyne

Under the supervision of

Dr Ashish Anand

Department of Computer Science & Engineering
INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI
Guwahati - 781039, India

July 11, 2020

Acknowledgements

I am indebted to my thesis examiners – Dr Biplab Bose (Chairperson), Prof. Sushmita Mitra (external examiner, Indian Statistical Institute, Kolkata, India - 700108), Prof. Luonan Chen (external examiner, Key Laboratory of Systems Biology, China - 200031), Prof. Pradip Kr. Das, Dr Amit Awekar, Dr Srinivasan Krishnaswamy, and Dr Prithwijit Guha – for their valuable suggestions; to my teachers at IIT Guwahati for strengthening my foundation; to my colleagues for making me a part of their community; to non-teaching staffs for enabling me to focus my entire energy on this thesis; to my research group members for complementing my knowledge with theirs; to my friends and family for nurturing my inner spirit; finally, to my Thesis Supervisor, Dr Ashish Anand – without whom the thesis would not exist, for helping me find a direction in life and initiating me in that direction.

December, 2019 Saptarshi Pyne

b

Declaration

I certify that

- The work contained in this thesis is original and has been done by myself and under the general supervision of my supervisor.
- The work reported herein has not been submitted to any other Institute for any degree or diploma.
- Whenever I have used materials (concepts, ideas, text, expressions, data, graphs, diagrams, theoretical analysis, results, etc.) from other sources, I have given due credit by citing them in the text of the thesis and giving their details in the bibliography. Elaborate sentences used verbatim from published work have been identified and quoted.
- I also affirm that no part of this thesis can be considered plagiarism to the best of my knowledge. I understand and take full responsibility if any complaint arises.
- I am fully aware that my thesis supervisor is not in a position to check for every possible instance of plagiarism within this submitted work.

December, 2019 Saptarshi Pyne

d

Certificate

This is to certify that this thesis titled 'Improving Reconstruction Efficiency of Time-varying Gene Regulatory Networks' submitted by Saptarshi Pyne, in partial fulfilment of the requirements for the award of the degree of Doctor of Philosophy, to the Indian Institute of Technology Guwahati, Assam, India, is a record of the bona fide research work carried out by him under my guidance and supervision at the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, Assam, India. To the best of my knowledge, no part of the work reported in this thesis has been presented for the award of any degree at any other institution.

Date: December, 2019
Place: Guwahati, India

Dr Ashish Anand (Thesis Supervisor)
Email: anand.ashish@iitg.ernet.in
Associate Professor,
Department of Computer Science and Engineering
Indian Institute of Technology Guwahati
Guwahati, India - 781039.

Abstract

Complex biological processes, such as - ageing and cancers, progress over time through multiple levels of biological regulations. For example, the progression of a cancer may involve mutations at the DNA-sequence level, followed by changes at the gene-regulation and protein-expression levels, and ultimately uncontrolled cell-growths at the tissue level. Comprehending such a process at each of its levels and their inter-level connections is a difficult task. Hence, attempts are made to understand each of its levels separately. Understanding a process at the gene-regulation level is the motivation for this thesis.

At the gene-regulation level, genes regulate each others' expression levels. If a gene's expression is regulated by that of one or more genes during a time interval, then it is said that the former gene is regulated by the latter during that time interval. To decipher such regulator-regulatee relationships, a two-step approach is traditionally followed. First, expressions of the concerned genes are collected across multiple time points. Second, that time-series gene expression data is analysed to reverse-engineer the underlying relationships. When the data contains a large number of genes, the reverse-engineering task necessitates the help of computational algorithms.

The said algorithms model the underlying relationships as a temporally-ordered sequence of directed networks. In each network, the nodes represent the genes, and every directed edge represents the 'regulator $\rightarrow$ regulatee' relationship between two genes. The i^{th} network in the modelled sequence describes the relationships during the i^{th} time interval. The algorithms produce the modelled sequence as output, which is known as the 'time-varying gene regulatory networks'; and the modelling process is known as the 'reconstruction' of time-varying gene regulatory networks. The objective of this thesis is to overcome some of the challenges faced by computational algorithms in the reconstruction of time-varying gene regulatory networks.

To identify the challenges, a comparative study is conducted between the existing reconstruction algorithms. For this study, three widely-used benchmark datasets with 10, 50 and 100 genes are utilised. It is observed that an algorithm named 'Auto Regressive TIme VArying models', in short, *ARTIVA*, outperforms all other algorithms in terms of correctness of the reconstructed networks. The correctness is measured with a widely-used metric called 'F1-score'; it represents how good an algorithm is in capturing the correct edges as well as rejecting the incorrect edges. Although *ARTIVA* demonstrates superiority in F1-score, it lacks in computational efficiency. *ARTIVA* consumes almost 1.5 days for the 100-gene benchmark dataset. With that speed, *ARTIVA* may require months for large-scale datasets that contain hundreds to thousands of genes. Such long time frames make the application of *ARTIVA* prohibitive with large-scale datasets. Thus, the objective of this thesis is further refined to develop algorithms that meet the following requirements:

- deliver correctness competitive to that of *ARTIVA*,
- offer computational efficiency compatible with large-scale datasets that contain hundreds to thousands of genes.

Towards that objective, we propose our first algorithm named 'an algorithm for reconstructing Time-varying Gene regulatory networks with Shortlisted candidate regulators', in short, *TGS*. This algorithm outpaces *ARTIVA* for all benchmark datasets. However, *ARTIVA* retains its superiority in F1-score.

To enhance F1-scores, we propose our second algorithm named 'TGS-Plus' (*TGS+*). This algorithm supersedes *ARTIVA* in F1-score. Moreover, *TGS+* outpaces *TGS* in runtime. For the 100-gene benchmark dataset, it consumes only 1 minute. With such time-efficiency and correctness, *TGS+* can be extremely suitable for large-scale datasets. However, that is not the case. We observe that both *TGS* and *TGS+* are unable to manage computational memory efficiently. Their memory requirements grow exponentially with the number of genes. This is a major concern for large-scale datasets.

To improve memory-efficiency, we propose our third set of algorithms. It contains two algorithms. The first one is called 'TGS - which is Light on memory', in short, *TGS-Lite*. This algorithm offers the same correctness and time-efficiency as that of *TGS*, yet, its memory requirement grows only linearly with the number of genes. Similarly, the second algorithm, known as 'TGS-Lite Plus' (*TGS-Lite+*), delivers the same correctness and time-efficiency as that of *TGS+*, at a linear memory requirement. Nonetheless, it is found that these algorithms, along with the ones we proposed earlier, tend to fail in capturing the edges that remain active for short periods of time. Such edges, known as 'transient edges', may have crucial effects. Hence, capturing transient edges is critical for understanding the underlying gene-regulation process.

For capturing transient edges, we propose our fourth and final set of algorithms. It consists of four algorithms. The first algorithm is named 'TGS - having Time-varying shortlists', in short, *TGS-T*. This algorithm captures significantly more numbers of edges than that of *TGS*. Similarly, the last three algorithms, named as {*TGS-T+*, *TGS-T-Lite*, *TGS-T-Lite+*} capture considerably more numbers of edges than that of {*TGS+*, *TGS-Lite*, *TGS-Lite+*}, respectively.

Overall, we propose four sets of algorithms in this thesis.

- The proposed algorithms advance the state of correctness for the reconstruction of time-varying gene regulatory networks.
- At the same time, they advance the state of computational efficiency, making them compatible with large-scale datasets.

Contents

1 Introduction **1**
1.1 Contributions of the Thesis . . . 3
1.2 Organisation of the Thesis . . . 9

2 Motivation **11**
2.1 Important Questions . . . 11
2.2 How to Answer . . . 12
2.2.1 How to Answer: The Systems Biology Approach . . . 12
2.3 Chapter Summary . . . 15

3 Literature Survey **17**
3.1 Co-expression Based Models . . . 17
3.1.1 Input Data: Single system, Multiple time points . . . 17
3.1.2 Correlation Networks . . . 17
3.1.3 Information Theoretic Models . . . 19
3.2 Conditional Independence (CI) Models . . . 21
3.2.1 Full Conditional Independence (CI) Models / Markov Random Fields (MRFs) / Markov Networks (MNs) . . . 22
3.2.2 Low Order Conditional Independence (CI) Models . . . 24
3.2.3 Bayesian Network (BayesNet) . . . 24
3.3 Joint Network Inference (JNI) Models . . . 26
3.3.1 Input Data: Single system, Multiple conditions, Multiple time points per condition . . . 27
3.3.2 Independent Network Inference (INI) with Gaussian Graphical Models (GGMs) . . . 27
3.3.3 Joint Network Inference (JNI) with Gaussian Graphical Models (GGMs) . . . 28
3.3.4 Joint Network Inference (JNI) for Joint Estimation of Multiple Related Systems . . . 34
3.4 Chapter Summary: Abridged Literature Survey . . . 39

4 Problem Formulation **45**
4.1 Notations . . . 45
4.1.1 Input: Time-series Gene Expression Dataset . . . 45
4.1.2 Output: Time-varying Gene Regulatory Networks . . . 45
4.2 Benchmark Datasets . . . 47
4.3 Evaluation Metrics . . . 48
4.4 Comparative Study of the Existing Algorithms . . . 48
4.4.1 Implementations . . . 49
4.4.2 Results . . . 49
4.5 Problem Statement . . . 49

4.6 Chapter Summary 50

5 Improving Time-efficiency **51**
5.1 Methods 51
5.1.1 Development of the Baseline Algorithm 51
5.1.2 Development of a Novel Algorithm: The TGS Algorithm (short form for 'an algorithm for reconstructing Time-varying Gene regulatory networks with Shortlisted candidate regulators') 52
5.2 Results 56
5.2.1 Discretisation of the Datasets 56
5.2.2 Implementations 56
5.2.3 Learning From Dataset Ds10n 56
5.2.4 Learning From Datasets Ds50n and Ds100n 57
5.2.5 Effects of Noise on Learning Power and Speed 60
5.3 Excerpt and Future Work 60
5.4 Contributions 62
5.5 Chapter Summary 62

6 Balancing Recall and Precision **63**
6.1 Methods 63
6.2 Results 64
6.2.1 Implementations 64
6.2.2 Learning from the Benchmark Datasets 64
6.3 Excerpt and Future Work 65
6.4 Contributions 66
6.5 Chapter Summary 66

7 Improving Memory-efficiency **69**
7.1 Task at Hand: Revisited 69
7.2 Limitations of the Previously Proposed Algorithms 70
7.3 Investigations into the Origin of the Limitations 72
7.4 A Novel Idea for Overcoming the Limitations 72
7.5 Design of Novel Algorithms Based on the Novel Idea 74
7.6 Experimental Results 74
7.6.1 Comparative Study Against a Random Classifier 74
7.6.2 Comparative Study Against Alternative Algorithms 76
7.6.3 Comparative Study Against Time-invariant Algorithms 80
7.6.4 Results with a Large-scale Dataset 87
7.7 Excerpt and Future Work 90
7.8 Contributions 92
7.9 Chapter Summary 92

8 Capturing Transient Edges **93**
8.1 Limitations of the Previously Proposed Algorithms 93
8.2 A Novel Idea for Overcoming the Limitations 94
8.3 Design of Novel Algorithms Based on the Novel Idea 94
8.3.1 The Issue with Extending *TGS+* and *TGS-Lite+* 97
8.3.2 Developing a Time-varying Refinement Strategy 99
8.3.3 Section Summary 102
8.4 Experimental Setup 104
8.4.1 Evaluation Strategy 104
8.4.2 Implementations 105

8.5 Results and Discussions . . . 105
8.5.1 Comparative Study Against Alternative Algorithms . . . 105
8.5.2 Additional Comparative Study Against a Random Classifier . . . 109
8.6 Excerpt and Future Work . . . 109
8.7 Contributions . . . 110
8.8 Chapter Summary . . . 110

9 Conclusions and Future Directions **113**
9.1 Summary of the Contributions . . . 113
9.2 Limitations and Future Directions . . . 114

Bibliography **119**

Publications **127**

A Pseudocodes **129**

List of Figures

1.1 Differences in Memory Usage between *TGS* and *TGS-Lite*. The values represent percentages of memory used, where the total memory is 32 GB in size. 7

2.1 **Network Reconstruction.** For a system of interest, values of its variables are measured across time and under different conditions. These measurements are inputted into a reconstruction algorithm that outputs a network model of the concerned system. For each class of network models, multiple reconstruction algorithms can be designed to suit different types of measurements. 15

3.1 Input data matrix $\mathbf{D}$. Given a system under observation, let us consider that $\mathbf{D}$ is a $(p \times N)$ matrix. It contains measurements of p system components across N samples. Different samples may be collected at distinct time points or from different tissues or under varied conditions. When each sample is collected at a distinct time point, it constitutes a time-series data with N time points. Suppose that V denotes the set of system components. Each system component $v \in V$ is modelled as a random variable X_v. Also, all system components together are modelled as a random vector $\mathbf{X} = (X_1, \ldots, X_p)$. Therefore, dataset $\mathbf{D}$ contains N realizations of $\mathbf{X}$, denoted as $\left(x^{(1)}, \ldots, x^{(N)}\right)$. An example of $\mathbf{D}$ can be a gene expression dataset having measurements of $p = 20,000$ genes across $N = 30$ time points from a single cell or averaged from a collection of cells. 18

3.2 **Reconstruction of Correlation Network.** This figure is reproduced from Villa-Vialaneix 2014, slide 7. It provides a visual representation of how Algorithm 1 works. First, pairwise correlations are calculated and stored in a $(p \times p)$ matrix (the leftmost sub-figure), where p = number of variables. The value at $(i, j)^{th}$ cell represents the Pearson correlation coefficient between the variables i and j. All the cell values are reduced to zero if it is less than a predefined threshold (sub-figure in the middle). Finally the matrix is visualized as an undirected graph (the rightmost sub-figure) where absence of edge between two nodes i and j implies that they are uncorrelated i.e. value at $(i, j)^{th}$ cell is zero. It can be noted that Pearson correlation coefficient is symmetric i.e. the values in the cells (i, j) and (j, i) of the correlation matrix should have the same value. But in this figure, the correlation matrix does not look exactly symmetric, which must be an unintentional mistake. 19

3.3 Input Data Tensor $\mathbf{D}$: Given a system under observation, let us assume that input data $\mathbf{D}$ is a $(p \times n_c \times k)$ tensor. It contains measurements of p system components under k different conditions. For each condition c, there are n_c independent observations, suppose at n_c different time points. Total number of observations is $\sum_{c=1}^{k} n_c = N$. Each system component $v \in V$ is modelled as k random variables $\{X_v^{(c)} : \text{For all } c \in [k]\}$ corresponding to k different conditions; $[n]$ stands for the set of first n natural numbers starting from 1. Then all the system components can be modelled as $(p \times k)$ random variables $\{X_j^{(c)} : j \in [p]\,,\ c \in [k]\}$. Such a dataset is very common in interventional experiments. Given a system, such as a cancer patient, k tissue samples are collected from the tumour. Then each of them are treated (intervened) with a different drug. Post intervention, gene expression of the same $p = 20,000$ genes are measured across multiple time points under each of the conditions. Being able to infer the changes in genetic dependency structures of tumour cells in response to different types of treatment, may help the researchers to identify the best treatment strategy against the cancer. It is to note that different conditions may also refer to different tissues from where the data is collected. Such spatio-temporal dataset can be very helpful to understand the mechanism behind spread of a disease across different parts of the body. 26

3.4 **Difference between INI and JNI strategies.** The whole input dataset is denoted by $\mathbf{D}$. $\mathbf{D}^{(c)}$ represents the sub-dataset specific to the condition c. Similarly, $G^{(c)}$ stands for the reconstructed graphical model of condition c. INI uses only the sub-dataset $\mathbf{D}^{(c)}$ to reconstruct $G^{(c)}$. On the other hand, JNI takes the whole dataset $\mathbf{D}$ as input and reconstructs all $G^{(c)}$s simultaneously through information sharing. 29

3.5 Input Data Tensor $\mathbf{D}$. Given I number of related biological systems under observation, let us assume that input data $\mathbf{D}$ is a $(p \times n_{c;i} \times k \times I)$ tensor. Subset of the data corresponding to a particular individual system i is denoted by $\mathbf{D}_i$. The relationship between the given systems would be defined on a case-by-case basis. Each $\mathbf{D}_i$ contains measurements of p system components under k different conditions. For each condition c, there are $n_{c;i}$ independent observations, suppose at $n_{c;i}$ different time points. Total number of observations in $\mathbf{D}_i$ is $\sum_{c=1}^{k} n_{c;i} = N_i$. Similarly, the total number of observations in $\mathbf{D}$ is $\sum_{i=1}^{I} N_i = N$. The system components $v \in V$ are modelled as k random variables $\{X_v^{(c)} : \text{For all } c \in [k]\}$ corresponding to k different conditions; $[n]$ stands for the set of first n natural numbers starting from 1. Then for each of the systems, all its components can be modelled as $(p \times k)$ random variables $\{X_j^{(c)} : j \in [p]\,,\ c \in [k]\}$. Such dataset is typically generated in cohort cell-line studies where k replicates (copies) of diseased tissues are collected from each of the I patients suffering from the same disease. Then each of the patient-specific tissue replicates is treated with a distinct drug. The effect of the drug is monitored for multiple time points. 35

3.6 Dynamic Bayesian Network (DBN) Modelling. First, an unrolled DBN is inferred for each system $i \in [I]$. The measurements of variable $j1$ in system i at time point $(t-1)$ is represented as node $\mathbf{D}_{i;j1}$ at time point $(t-1)$ (hereafter, $\mathbf{D}_{i;j1}(t-1)$). An edge from $\mathbf{D}_{i;j1}(t-1)$ to $\mathbf{D}_{i;j2}(t)$ implies that the observations of $j2$ at time point t is not conditionally independent of that of $j1$ at the previous time point. After the unrolled DBN reconstruction is completed, it is summarised into a static DBN following some rule, for example - add an edge $(j1, j2)$ in static G_i if there is an edge $(\mathbf{D}_{i;j1}, \mathbf{D}_{i;j2})$ in at least 80% of the temporal transitions (total number of temporal transitions = total number of time points − 1). For example, the edge $(j2, j3)$ is not added to the static DBN because the edge $(\mathbf{D}_{i;j2}, \mathbf{D}_{i;j3})$ (the red arrow) appears only in 50% of the total transitions in the unrolled DBN. It can be noted that static DBN allows cycles unlike BayesNet. 37

3.7 A graphical summary of the literature review. A single-bordered white box indicates that the corresponding method generates undirected network using an Independent Network Inference (INI) algorithm. On the other hand, the double-border signifies that a Joint Network Inference (JNI) algorithm is employed. A blue box implies that the generated network or networks is a directed graph. A directed edge between two boxes signifies that the end box is an extension of the start box. Reference to each publication is given in Table 3.1. 40

4.1 Input time-series gene expression data $\mathcal{D}$ is a three dimensional tensor with the dimensions (V genes, T time points, S time series). $\mathcal{D}$ is comprised of a set of time series $\mathcal{S} = \{s_1, \ldots, s_S\}$ of gene expression data. Each time series contains the expression levels of a set of genes $\mathcal{V} = \{v_1, \ldots, v_V\}$ at T consecutive time points $\mathcal{T} = \{t_1, \ldots, t_T\}$. It is assumed that there are no missing values in any time series. In other words, each time series is a complete time series of T time points. Notation $\mathcal{D}_{(\mathcal{X};\mathcal{Y};\mathcal{Z})}$ is used to denote the observed values of genes $\mathcal{X}$ at time points $\mathcal{Y}$ in time series $\mathcal{Z}$. Hence, $\mathcal{D}_{(\mathcal{X};\mathcal{Y};\mathcal{Z})} \subseteq \mathcal{D}$ where $\mathcal{X} \subseteq \mathcal{V}, \mathcal{Y} \subseteq \mathcal{T}, \mathcal{Z} \subseteq \mathcal{S}$. . 46

4.2 Output time-varying GRNs $\left(G^{(1)}, \ldots, G^{(T-1)}\right) = \mathcal{G}$ is a sequence of directed unweighted networks. Here, $G^{(p)} (\in \mathcal{G})$ represents the gene regulatory events occurred during the time interval between time points t_p and $t_{(p+1)}$. It consists of $(2 \times V)$ nodes $\{v_i\_t_q : v_i \in \mathcal{V}, t_q \in \{t_p, t_{(p+1)}\}\}$. There exists a directed unweighted edge $\left(v_i\_t_p, v_j\_t_{(p+1)}\right)$ if and only if v_i regulates v_j during time interval $\left(t_p, t_{(p+1)}\right)$. 47

5.1 Graphical Flowchart (Part 1) of the *TGS* Algorithm. The flowchart is continued in Figure 5.2. For illustration, a dataset $\mathcal{D}$ is considered with four genes $\{v_1, v_2, v_3, v_4\} = \mathcal{V}$ and two time series $\{S_1, S_2\} = \mathcal{S}$. Each time series has three time points $\{t_1, t_2, t_3\} = \mathcal{T}$. $\mathcal{D}$ is discretised into two discrete levels, represented by $\{1, 2\}$. 54

5.2 Graphical Flowchart (Part 2) of the *TGS* Algorithm. The flowchart is continued from Figure 5.1. For discussion of the *Bene* step, let us consider the 'Selection of regulators of $\{v_1\_t_{(p+1)} : 2 \leq (p+1) \leq T\}$'. Since, v_2 is the sole neighbour of v_1 in G_{CLR}, v_2 is the only candidate regulator of v_1 (Figure 5.1). Therefore, the candidate regulator sets of $v_1\_t_2$ are $\emptyset$ and $\{v_2\_t_1\}$. Among these two sets, *Bene* chooses $\{v_2\_t_1\}$ based on observations $\mathcal{D}_{\mathcal{V};\{t_1,t_2\};\mathcal{S}}$. Similarly, the candidate regulator sets of $v_1\_t_3$ are $\emptyset$ and $\{v_2\_t_2\}$. Among these two sets, *Bene* chooses $\emptyset$ based on observations $\mathcal{D}_{\mathcal{V};\{t_2,t_3\};\mathcal{S}}$. 55

5.3 Runtime of the *TGS* Algorithm w.r.t. the Number of Genes in the Benchmark Datasets (Table 4.1). The black and grey lines represent noisy and noiseless versions of the datasets, respectively. 59

5.4 Precision of the *TGS* Algorithm w.r.t. the Number of Genes in the Benchmark Datasets. The black and grey bars represent noisy and noiseless versions of the datasets, respectively. The *2L.wt* algorithm is used for data discretisation. 60

7.1 The Workflow of a Time-varying GRN Reconstruction Algorithm. The algorithm takes a time-series gene expression data $\mathcal{D}$ as input. The data consists of S number of time series. Each time series contains measured expressions of V number of genes across T number of time points. In return, the algorithm outputs time-varying GRNs $\left(G^{(1)}, \ldots, G^{(T-1)}\right) = \mathcal{G}$, which is a sequence of directed unweighted networks. Here, $G^{(p)} (\in \mathcal{G})$ represents the gene regulatory events occurred during the time interval between time points t_p and $t_{(p+1)}$. It consists of $(2 \times V)$ nodes $\{v_i\_t_q : v_i \in \mathcal{V}, t_q \in \{t_p, t_{(p+1)}\}\}$. There exists a directed unweighted edge $\left(v_i\_t_p, v_j\_t_{(p+1)}\right)$ if and only if v_i regulates v_j during time interval $\left(t_p, t_{(p+1)}\right)$. For instance, $G^{(1)}$ represents the regulatory events that occurred between time points t_1 and t_2. One such event is the regulatory effect of $v_1\_t_1$ on $v_2\_t_2$ as represented by a directed edge. 71

7.2 Illustration of the Idea for Finding the Highest Scoring Subset with $2^{|\mathcal{V}_{(j;(p+1))}|}$ Subsets, Two Scores and Two Pointers. Each rectangle represents a location in the memory. It is assumed that $\mathcal{V}_{(j;(p+1))} = \{v_{i_1}\_t_p, v_{i_2}\_t_p\}$. Therefore, its subsets are $\{\emptyset, \{v_{i_2}\_t_p\}, \{v_{i_1}\_t_p\}, \{v_{i_1}\_t_p, v_{i_2}\_t_p\}\}$. Each subset is represented as a binary string of Boolean TRUE (1_b) and FALSE (0_b) values, indicating whether a gene is present or not in that subset, respectively. All the subsets are held in memory simultaneously as depicted by the four-row two-column table in the middle. In addition, two pointers ('curr.set' and 'best.set') and two scores ('curr.score' and 'best.score') are also stored in memory (each score is a floating point number). In step 1, the first subset is pointed to by curr.set (current subset). In step 2, the score of this subset is calculated and stored inside curr.score (current subset's score). Subsequently, the values of curr.set and curr.score are copied to best.set (hitherto best subset) and best.score (hitherto best score), respectively. In step 3, the second subset is pointed to by curr.set. In step 4, its score is calculated and stored inside curr.score. If curr.score is greater than best.score, then the values of best.score and best.set are replaced with that of curr.score and curr.set, respectively. Otherwise, if curr.score is less than or equal to best.score, then best.score and best.set are kept unchanged. Next in step 5, the third subset is pointed to by curr.set and the same procedure is followed till all the subsets are exhausted. As a result, the final value of best.set points to the subset with the highest score (ties between two equally scoring subsets are broken in favour of the subset with the lower index). Overall, this strategy requires two pointers (curr.set and best.set), two scores (curr.score and best.score) and $2^{|\mathcal{V}_{(j;(p+1))}|} = 2^2 = 4$ subsets to be held in memory. It also requires corresponding gene expression data (not shown in figure) to be stored in memory for calculating the scores. 73

7.3 Illustration of the Idea for Finding the Highest Scoring Subset amongst $2^{|\mathcal{V}_{(j;(p+1))}|}$ subsets with Only Two Subset-variables, Two Scores and One Subset-generation Script. Each solid-lined rectangle represents a location in the memory. It is assumed that $\mathcal{V}_{(j;(p+1))} = \{v_{i_1}\_t_p, v_{i_2}\_t_p\}$. Therefore, its subsets are $\{\emptyset, \{v_{i_2}\_t_p\}, \{v_{i_1}\_t_p\}, \{v_{i_1}\_t_p, v_{i_2}\_t_p\}\}$. Each subset is represented as a binary string of Boolean TRUE (1_b) and FALSE (0_b) values, indicating whether a gene is present or not in that subset, respectively. Initially, the empty subset is stored in location 'curr.set' (current subset). Please note that curr.set holds the subset itself and not a pointer to the subset. In step 1, the score of the empty subset is calculated and stored inside 'curr.score' (current subset's score which is a floating point number). Subsequently, the values of curr.set and curr.score are copied into 'best.set' (hitherto best subset) and 'best.score' (hitherto best score), respectively. In step 2, the value of curr.set is sent to the subset-generation script, namely 'GEN-NEXT-SET'. In step 3, the script generates the next value of curr.set i.e. the second subset (indicated by the dotted-lined rectangle). In step 4, the score of the second subset is calculated and stored inside curr.score. If curr.score is greater than best.score, then the values of best.score and best.set are replaced with that of curr.score and curr.set, respectively. Otherwise, if curr.score is less than or equal to best.score, then best.score and best.set are kept unchanged. In step 5, the value of curr.set is sent to the script, which generates the next value of curr.set in step 6. The same procedure is followed till all the subsets are exhausted. As a result, the final value of best.set represents the highest scoring subset (ties between two equally scoring subsets are broken in favour of the subset with lower index). Overall, this strategy requires two subset-variables (curr.set and best.set), two scores (curr.score and best.score) and one subset-generation script (GEN-NEXT-SET) to be held in memory. It also requires corresponding gene expression data (not shown in figure) to be stored in memory for calculating the scores. 75

7.4 The TPR-vs-FPR Plots of *TGS-Lite*, *TGS-Lite+* and a random classifier. Here, TPR = True Positive Rate, FPR = False Positive Rate. The results of *TGS-Lite* for three distinct benchmark datasets are represented as three black squares. These squares are connected (interpolated) with a smooth line (Software used: LibreOffice Calc Version 5.1.6.2; Line type = Cubic spline, Resolution = 20; OS: Ubuntu 16.04.5 LTS). On the other hand, the results of *TGS-Lite+* for three distinct benchmark datasets are represented as three black triangles (the triangle for Ds10n overlaps with the square for Ds10n and hence not visible). These triangles are also connected with a smooth line. 76

7.5 Comparative Memory Requirements of the *TGS* and *TGS-Lite* Algorithms. The percentage of memory usage ('%MEM') by *TGS* and *TGS-Lite* when the max fan-in parameter is set to 24. The shown figure is a screenshot of the 'top' command in the Ubuntu OS. 77

7.6 Comparative Performance of the Selected Algorithms on the Benchmark Datasets. **A)** Recall, **B)** precision and **C)** F1-scores of the selected algorithms are shown. 78

7.7 The Effects of the Multicore Parallelisation on the Runtime of *TGS-Lite* and *TGS-Lite+*. Dataset Ds100n is used and the max fan-in parameter is set to 14. 79

7.8 The Effects of the Max Fan-in Parameter on the Correctness (**A, B**) and Runtime (**C, D**) of the TGS-Lite and TGS-Lite+ Algorithms. Dataset Ds100n is used and number of cores is set to 10 for multicore parallelisation. 91

8.1 An Example for Illustrating the Limitations of the Previously Proposed Algorithms. In this example, a system with four genes – $\{v_1, \cdots, v_4\}$ – is considered. Among them, we arbitrarily choose v_2 whose regulator(s) we want to predict. The true regulator(s) of v_2 during different time intervals are shown with the directed edges, across five time points – $\{t_1, \cdots, t_5\}$. We assume that these relationships are reflected in a time-series dataset. Given that dataset, the previously proposed algorithms generate a short-list of candidate regulators. Since, v_2 itself and v_4 have no regulatory relationship with v_2 in any of the time intervals, they are expected to share a low mutual information with v_2; thus, they are correctly rejected. On the other hand, v_1 is expected to share a high mutual information with v_2 during the first time interval and low during all other time intervals. Therefore, when mutual information are computed from the whole dataset, v_1 is less likely to be shortlisted. However, v_3 who is expected to share high mutual information with v_2 across most time intervals, is most likely to get shortlisted. If v_1 is not shortlisted, it is definitely not going to appear in the final list. Thus, the previously proposed algorithms are not able to capture the transient edge from gene v_1 to v_2 during time interval (t_1, t_2). 95

8.2 An Example for Illustrating the Idea of Time-varying Short-listing. In this example, a system with four genes – $\{v_1, \cdots, v_4\}$ – is considered. Among them, we arbitrarily choose v_2 whose regulator(s) we want to predict. The true regulator(s) of v_2 during different time intervals are shown with the directed edges, across five time points – $\{t_1, \cdots, t_5\}$. We assume that these relationships are reflected in a time-series dataset. Given that dataset, one shortlist of candidate regulators is generated for each time interval. Each shortlist is expected to capture the true regulator(s) during the corresponding time interval, such as v_1 is shortlisted during (t_1, t_2); it may also incorrectly capture some false regulator(s), like v_2 itself is shortlisted during the same time interval. For each time interval, the final selection step uses the data and shortlist, both specific to that interval, and predicts the final set of regulators. Since, the transient regulators are captured in the shortlists, they are highly likely to be captured in the final lists. Thus, the idea of time-varying short-listing is likely to capture more transient edges than that of the previous strategy of time-invariant short-listing. 96

8.3 Illustration of the Modified Framework with an Example. In this example, the input dataset is comprised of two time series – s_1 and s_2. Each time series contains the expressions of four genes $\{v_1, \cdots, v_4\}$ across three time points $\{t_1, \cdots, t_3\}$. The dataset is discretised as required by the proposed framework. Given this dataset, step 1 of the modified framework generates time-varying shortlists of candidate regulators. Consequently in Step 2, the final sets of regulators are chosen from the corresponding shortlists. 98

8.4 Illustration of the Time-invariant Refinement strategy by the *ARACNE* Algorithm. The strategy is explained with an example. In this example, we assume that two genes – v_i and v_j share a non-zero mutual information, denoted by $\mathscr{M}(v_i, v_j)$. It is represented with an undirected edge since mutual information is symmetric in nature i.e. $\mathscr{M}(v_i, v_j) = \mathscr{M}(v_j, v_i)$. Therefore, *ARACNE* needs to figure out whether this mutual information is due to a direct regulatory relationship between the two genes or not. For that purpose, ARACNE searches for a third gene v_k that satisfies the condition: $\mathscr{M}(v_i, v_j) < \min(\mathscr{M}(v_i, v_k), \mathscr{M}(v_k, v_j))$. If *ARACNE* is able to find at least one such v_k, it assumes that v_i and v_j do not have a direct regulatory relationship. Instead, they are related through v_k. One such case could be where v_i regulates v_k and v_k in turn regulates v_j. In such cases, *ARACNE* sets $\mathscr{M}(v_i, v_j) = 0$. On the other hand, if *ARACNE* is unable to find any such v_k, it assumes that v_i and v_j have a direct regulatory relationship. Therefore, the value of $\mathscr{M}(v_i, v_j)$ is kept unchanged. Thus, *ARACNE* checks and refines (if necessary) the mutual information values between every pair of genes. 99

8.5 An Example of the DPI. In this example, gene v_i regulates gene v_k and v_k in turn regulates gene v_j. No other regulatory relationships exist between v_i and v_j. In that case, the DPI states that the following inequality must be satisfied: $\mathscr{M}(v_i, v_j) \leq \min(\mathscr{M}(v_i, v_k), \mathscr{M}(v_k, v_j))$. 100

8.6 A Realistic Case where Conjecture 1 Holds. In this case, $v_i\_t_p$ and $v_k\_t_p$ are co-regulated by the same gene, suppose, $v_l\_t_{(p-1)}$. Therefore, the formers are highly likely to share a non-zero mutual information i.e. $\mathscr{M}(v_i\_t_p, v_k\_t_p) > 0$. On the other hand, $v_k\_t_p$ regulates $v_j\_t_{(p+1)}$. Hence, $v_k\_t_p$ and $v_j\_t_{(p+1)}$ also share a non-zero mutual information i.e. $\mathscr{M}\left(v_k\_t_p, v_j\_t_{(p+1)}\right) > 0$. Since $v_i\_t_p$ shares a non-zero mutual information with $v_k\_t_p$ which again shares a non-zero mutual information with $v_j\_t_{(p+1)}$, $v_i\_t_p$ might share a non-zero mutual information with $v_j\_t_{(p+1)}$ i.e. $\mathscr{M}\left(v_i\_t_p, v_j\_t_{(p+1)}\right) > 0$. However, according to the DPI, this false mutual information must not exceed the true mutual informations that have caused its non-zeroness. In other words, $\mathscr{M}\left(v_i\_t_p, v_j\_t_{(p+1)}\right)$ must not exceed $\mathscr{M}(v_i\_t_p, v_k\_t_p)$ or $\mathscr{M}\left(v_k\_t_p, v_j\_t_{(p+1)}\right)$. 101

8.7 Illustration of the Workflow of *ARACNE-T* with an Example. In this example, the input dataset is disretised and comprised of two time series – s_1 and s_2. Each time series contains the expressions of four genes $\{v_1, \cdots, v_4\}$ across three time points $\{t_1, \cdots, t_3\}$. From this dataset, an ordered list of time-varying raw mutual information matrices is estimated. Each cell in a matrix contains a non-negative real number which represents a mutual information value (the numbers are not shown inside the cells to save space). For example, the shaded cell contains the mutual information between $v_2\_t_1$ and $v_3\_t_2$. This ordered list is given as input to *ARACNE-T* which in turn produces an equivalent list of refined mutual information matrices. 103

8.8 The Black Box Diagrams of Algorithms *TGS+* and *TGS-T+*. The latter (in right) differs from the former (in left) in two places: first, ARACNE is replaced with ARACNE-T; second, CLR is replaced with CLR-T. . . 104

8.9 The TPR-vs-FPR Plots of *TGS-T*, *TGS-T+* and a random classifier. Here, TPR = True Positive Rate, FPR = False Positive Rate. The results of *TGS-T* and *TGS-T+* for three benchmark datasets – Ds10n, Ds50n and Ds100n – are represented as three black squares. These squares are connected (interpolated) with a smooth line (Software used: LibreOffice Calc Version 5.4.7.2 (x64); Line type = Cubic spline, Resolution = 20; OS: Windows 10 Pro version 1809). On the other hand, the results of the random classifier is presented as a dashed line. 110

List of Tables

1.1 The Runtime of *TGS* and *ARTIVA* for the Benchmark Datasets. For each dataset, the fastest runtime is boldfaced. 4

1.2 The Recalls of *TGS* and *ARTIVA* for the Benchmark Datasets. For each dataset, the highest recall is boldfaced. 4

1.3 The F1-scores of *TGS* and *ARTIVA* for the Benchmark Datasets. For each dataset, the highest F1-score is boldfaced. 5

1.4 The Precision of *TGS* and *ARTIVA* for the Benchmark Datasets. For each dataset, the highest precision is boldfaced. 5

1.5 The F1-scores of *TGS+* and *ARTIVA* for the Benchmark Datasets. For each dataset, the highest F1-score is boldfaced. 6

1.6 The Runtime of *TGS+*, *TGS* and *ARTIVA* for the Benchmark Datasets. For each dataset, the fastest runtime is boldfaced. 6

1.7 The Recalls of *TGS-T* and *TGS* for the Benchmark Datasets. For each dataset, the highest recall is boldfaced. 8

3.1 References for Figure 3.7 41

4.1 A Summary of the chosen DREAM3 Datasets. Here, V = number of genes, T = number of time points, and S = number of time series. . . . 48

4.2 The F1-scores of the Existing Reconstruction Algorithms for Benchmark Datasets Ds10n, Ds50n and Ds100n. The numerical values are rounded off to three decimal places. For each dataset i.e. column, the best value is boldfaced. 49

4.3 The runtime of the Existing Reconstruction Algorithms for Benchmark Datasets Ds10n, Ds50n and Ds100n. The numerical values are rounded off to three decimal places. 50

5.1 Learning Power of the Selected Algorithms on Dataset Ds10n. TP = True Positive, FP = False Positive. Two ordered values in each cell for rows 'TBN' and 'TGS' represent application of two different data discretisation algorithms – *2L.wt* and *2L.Tesla*, respectively. On the other hand, other rows have a single value in each cell, since other algorithms do not require the dataset to be discretised. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced. . 57

5.2 Runtime of the Selected Algorithms on Dataset Ds10n. Two ordered values in each cell for rows 'TBN' and 'TGS' represent application of two different data discretisation algorithms – *2L.wt* and *2L.Tesla*, respectively. On the other hand, other rows have a single value in each cell, since other algorithms do not require the dataset to be discretised. In *TGS*, the *CLR* step takes 0.003 seconds for *2L.wt* and *2L.Tesla* each. . . 58

5.3 Learning Power of the Selected Algorithms on Dataset Ds50n. TP = True Positive, FP = False Positive. Algorithm *2L.wt* is used for data discretisation in *TGS*. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced. 58
5.4 Learning Power of the Selected Algorithms on Dataset Ds100n. TP = True Positive, FP = False Positive. Algorithm *2L.wt* is used for data discretisation in *TGS*. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced. 59
5.5 Runtime of the Selected Algorithms on Datasets Ds50n and Ds100n. For *TGS*, algorithm *2L.wt* is used for data discretisation. The *CLR* step in *TGS* takes 0.005 and 0.013 seconds for Ds50n and Ds100n, respectively. 59
5.6 Maximum Number of Neighbours a Gene has in the *CLR* Network. Algorithm *2L.wt* is used for data discretisation. 60

6.1 Learning Power of the Selected Algorithms on Dataset Ds10n. TP = True Positive, FP = False Positive. Algorithm *2L.wt* is used for data discretisation in case of *TGS* and *TGS+*. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced. 65
6.2 Learning Power of the Selected Algorithms on Dataset Ds50n. TP = True Positive, FP = False Positive. Algorithm *2L.wt* is used for data discretisation in case of *TGS* and *TGS+*. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced. 65
6.3 Learning Power of the Selected Algorithms on Dataset Ds100n. TP = True Positive, FP = False Positive. Algorithm *2L.wt* is used for data discretisation in case of *TGS* and *TGS+*. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced. 65
6.4 Runtime of the Selected Algorithms on the Benchmark Datasets. Algorithm *2L.wt* is used for data discretisation in case of *TGS* and *TGS+*. . 66

7.1 Runtime of the Selected Algorithms on the Benchmark Datasets. 79
7.2 Maximum Number of Candidate Regulators of a Gene in the *TGS-Lite* and *TGS-Lite+* Algorithms for a given Dataset. These statistics are recorded when the max fan-in restriction is not applied. If the max fan-in restriction is applied, the numbers will be upper bounded by the value assigned to the max fan-in parameter. 80
7.3 The Time-invariant Algorithms Selected for the Comparative Study in Section 7.6.3 81
7.4 Reconstruction Correctness of the Selected Algorithms on Dataset Ds10n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced. 82
7.5 Reconstruction Correctness of the Selected Algorithms on Dataset Ds50n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced. 82
7.6 Reconstruction Correctness of the Selected Algorithms on Dataset Ds100n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced. 83
7.7 Runtime of the Selected Algorithms on the Benchmark Datasets. 84

7.8 Reconstruction Correctness of the Selected Algorithms on Dataset Ds10n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced. 85
7.9 Reconstruction Correctness of the Selected Algorithms on Dataset Ds50n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced. 85
7.10 Reconstruction Correctness of the Selected Algorithms on Dataset Ds100n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced. 85
7.11 Runtime of the Selected Algorithms on the Benchmark Datasets. 86
7.12 A Summary of the DmLc3 Sub-datasets. 88
7.13 Runtime of the Selected Algorithms on the DmLc3 Sub-datasets. In each column, the lowest runtime is boldfaced. 90

8.1 Mapping from the Original Algorithms to the Extended Algorithms. . . 104
8.2 Performances of the Selected Algorithms on Dataset Ds10n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced. . 106
8.3 Performances of the Selected Algorithms on Dataset Ds50n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced. . 107
8.4 Performances of the Selected Algorithms on Dataset Ds100n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced. . 108

Chapter 1

Introduction

Complex biological processes, such as - ageing and cancers, progress over time through multiple levels of biological regulations (Yu et al., 2013). For example, the progression of a cancer may involve mutations at DNA sequences, consequent changes at gene regulations, followed by modulations in protein expressions, resulting in uncontrolled cell growth and ultimately the formation of tumours. Comprehending such a process requires understandings of each of its levels and their inter-connections. This is an extremely difficult task. Hence, attempts are made to understand each of its levels separately (Alon, 2006). Understanding a process at the level of gene regulation is the motivation for this thesis.

To understand a process at the level of gene regulation, a two-step approach is traditionally followed: first, expressions of the concerned genes are collected across multiple time points; second, that time-series gene expression data is analysed to reverse-engineer the underlying gene regulations (Sanguinetti and Huynh-Thu, 2019). When the data collected in the first step is quite large, the second step necessitates the help of computational algorithms. The said algorithms model the underlying gene regulations as a temporally-ordered sequence of directed networks. In each network, the nodes represent the genes, and every directed edge represents the 'regulator $\rightarrow$ regulatee' relationship between two genes. The first network in the modelled sequence describes the relationships during the first time interval i.e. between the first and second time points; the second network describes the relationships during the second time interval i.e. between the second and third time points, and so on. The algorithms produce the modelled sequence as output, which is known as the 'time-varying gene regulatory networks'; and the modelling process is known as the 'reconstruction' of time-varying gene regulatory networks (Grzegorczyk and Husmeier, 2011). The objective of this thesis is to overcome some of the challenges faced by computational algorithms in the reconstruction of time-varying gene regulatory networks.

To identify the challenges, we conduct a comparative study between the existing reconstruction algorithms. For this study, three widely-used benchmark datasets are utilised (Marbach et al., 2010, 2009; Prill et al., 2010); they are called Ds10n, Ds50n and Ds100n since they contain time-series expressions of 10, 50 and 100 genes, respectively. It is observed that one particular algorithm substantially outperforms all other algorithms in correctness of reconstructed networks. The said algorithm is called 'Auto Regressive TIme VArying models', in short, *ARTIVA* (Lèbre et al., 2010). Although *ARTIVA* demonstrates superiority in correctness, it lacks in computational efficiency. *ARTIVA* consumes almost 1.5 days for the benchmark dataset with 100 genes. With that speed, *ARTIVA* may require months for large-scale datasets, which usually contain hundreds to thousands of genes (Zaas et al., 2009). Such long time frames make the application

of *ARTIVA* prohibitive with large-scale datasets. Thus, we refine the objective of this thesis to develop algorithms that meet the following requirements:

- deliver correctness competitive to that of *ARTIVA*,
- offer computational efficiency compatible with large-scale datasets that contain hundreds to thousands of genes.

Towards that objective, we propose our first algorithm named 'an algorithm for reconstructing Time-varying Gene regulatory networks with Shortlisted candidate regulators', in short, *TGS*. This algorithm outpaces *ARTIVA* for all benchmark datasets, with considerable margins. As an example, for the benchmark dataset with 100 genes, where *ARTIVA* takes around 1.5 days, *TGS* consumes only about 18 minutes. At the same time, *TGS* consistently captures higher numbers of correct edges than that of *ARTIVA*, for all benchmark datasets. On the other hand, *ARTIVA* succeeds to reject significantly higher numbers of incorrect edges than that of *TGS* for two of the three benchmark datasets. As a result, *ARTIVA* retains its superiority in correctness, which is measured with a widely-used cumulative metric of capturing correct edges and rejecting incorrect edges known as 'F1-score' (Liu et al., 2016).

To enhance F1-scores, we propose our second algorithm named 'TGS-Plus' (*TGS+*). This algorithm has an additional step compared to *TGS*. The additional step prevents a considerable number of incorrect edges from being captured. Due to this step, *TGS+* is able to reject competitive numbers of incorrect edges to that of *ARTIVA*, while capturing competitive numbers of correct edges to that of *TGS*. As a consequence, *TGS+* supersedes *ARTIVA* in F1-score. Moreover, *TGS+* overtakes even *TGS* in runtime. For the benchmark dataset with 100 genes, where *TGS* takes about 18 minutes, *TGS+* requires only 1 minute. With that time-efficiency, *TGS+* can be extremely suitable for large-scale datasets. However, that is not the case. We observe that both *TGS* and *TGS+* are unable to manage computational memory efficiently. Their memory requirements grow exponentially with the number of genes. This is a major concern with large-scale datasets.

To improve memory-efficiency, we propose our third set of algorithms. It contains two algorithms. The first one is called 'TGS - which is Light on memory', in short, *TGS-Lite*. This algorithm offers the same correctness and time-efficiency as that of *TGS*, yet, at a low memory requirement that grows linearly with the number of genes. Similarly, the second algorithm, known as 'TGS-Lite Plus' (*TGS-Lite+*), delivers the same correctness and time-efficiency as that of *TGS+*, at a linear memory requirement. Such computational efficiencies make *TGS-Lite* and *TGS-Lite+* most suitable for large-scale datasets. Nonetheless, it is found that these algorithms, along with the ones we proposed earlier, tend to fail in capturing a particular type of edges. These edges are known as 'transient edges'; they remain active for short periods of time, but may have crucial effects. Therefore, capturing transient edges is critical for understanding the underlying gene-regulation process.

For capturing transient edges, we propose our fourth and final set of algorithms. It consists of four algorithms. The first algorithm is named 'TGS - having Time-varying shortlists', in short, *TGS-T*. This algorithm captures significantly more numbers of edges than that of *TGS*. As an example, for the benchmark dataset with 100 genes, *TGS-T* captures 49 out of 166 correct edges, compared to 28 captured by *TGS*. In a similar manner, the last three algorithms, named as {*TGS-T+*, *TGS-T-Lite*, *TGS-T-Lite+*} capture considerably more numbers of edges than that of {*TGS+*, *TGS-Lite*, *TGS-Lite+*}, respectively.

To summarise, we propose four sets of algorithms in this thesis. The following observations are made from comparative studies conducted between the proposed algorithms and the existing ones:

- The proposed algorithms are significantly more time-efficient than the existing algorithms. It makes the former algorithms most suitable for processing large-scale datasets.
- At the same time, a subset of the proposed algorithms efficiently utilise computational memory, which makes them extremely suitable for processing large datasets when available memory is limited.
- Most importantly, the proposed algorithms do not offer the aforementioned efficiencies at the cost of correctness. On the contrary, they surpass the existing algorithms even in correctness.

Through the aforementioned contributions, this thesis advances the state of computational efficiency for the reconstruction task of time-varying gene regulatory networks. It also advances the state of correctness for the same. In the next section, we discuss the contributions in finer details.

1.1 Contributions of the Thesis

Contribution 1: Improving Time-efficiency

Performing the reconstruction task efficiently necessitates a thorough understanding of the task at hand. To gain that understanding, we study the existing reconstruction algorithms. It is learnt that the task can be seen as finding out which gene is regulated by which gene (or genes) and when. Therefore, the whole task can be decomposed into sub-tasks of finding out the regulators of a particular gene at different time intervals. Each of these sub-tasks can be further decomposed into atomic tasks of finding out the regulators of a particular gene at a specific time interval.

The time required for executing each atomic task increases with the number of candidate regulators of the concerned gene. Since, any gene can be a regulator of the said gene, including the gene itself (through a process known as auto-regulation), the number of candidate regulators becomes same as the total number of genes. For that reason, the existing reconstruction algorithms require significant computational time for datasets with large numbers of genes.

To mitigate the issue, we propose a time-efficient algorithm named 'an algorithm for reconstructing Time-varying Gene regulatory networks with Shortlisted candidate regulators', in short, *TGS* (Pyne et al., 2020). This algorithm consists of two consecutive steps: the short-listing step and the final selection step.

In the short-listing step, a shortlist of candidate regulators is prepared for the concerned gene. The short-listing is performed based on a criterion known as 'Context Likelihood of Relatedness' (Faith et al., 2007). This criterion decides whether the expressions of a candidate regulator convey a significant amount of information about the expressions of the concerned gene; formally, whether the candidate regulator shares a statistically significant 'mutual information' with the concerned gene, during all time intervals (Cover and Thomas, 2012). If a candidate regulator shares a statistically significant mutual information with the concerned gene, it is concluded that the expressions

of the latter are not independent of that of the former. Hence, that candidate regulator is short-listed for further examinations.

Further examinations are conducted in the final selection step. This step tests whether the expressions of the concerned gene, during a particular time interval, is independent of that of the candidate regulator, given the expressions of other candidate regulators at that time interval. This test is based on the idea of 'conditional independence' (Markowetz and Spang, 2007); it says that the expressions of the concerned gene can be independent of that of the candidate regulator, given some condition, even though they are not independent when that condition is not given. For example, if the concerned gene and the candidate regulator have a common regulator, then the expressions of the former genes may not be independent; however, they would be conditionally independent, given the expressions of the common regulator. The final selection step checks whether the expressions of the concerned gene are conditionally independent of that of the candidate regulator during a particular time interval, using the most rigorous test of conditional independence known as 'Bayesian networks' (Markowetz and Spang, 2007). If the expressions of the concerned gene is not conditionally independent of that of the candidate regulator during a specific time interval, then the latter is selected as a regulator of the former in the reconstructed network specific to that time interval.

The time required for the Bayesian network tests grows exponentially with the number of candidate regulators. However, due to the short-listing step, the tests need to be performed only for the short-listed candidate regulators. At the same time, the time required by the short-listing step itself grows only quadratically with the total number of genes. Hence, the time consumed by the short-listing step is overshadowed by the time it saves during the final selection step. As a result, *TGS* outpaces all existing algorithms, including *ARTIVA*, in runtime (Table 1.1).

Table 1.1: The Runtime of *TGS* and *ARTIVA* for the Benchmark Datasets. For each dataset, the fastest runtime is boldfaced.

Dataset	**TGS**	**ARTIVA**
Ds10n	**5.789 s**	10 m 20 s
Ds50n	**7 m 36 s**	4 h 30 m 15 s
Ds100n	**17 m 49 s**	31 h 52 m 54 s

Moreover, *TGS* consistently outperforms *ARTIVA* in capturing correct edges, which is measured with a metric named 'recall' (Table 1.2). Recall is the ratio of the number of correct edges in a model to the maximum achievable number of correct edges; hence, recall represents how good an algorithm is in capturing the correct edges (Liu et al., 2016).

Table 1.2: The Recalls of *TGS* and *ARTIVA* for the Benchmark Datasets. For each dataset, the highest recall is boldfaced.

Dataset	**TGS**	**ARTIVA**
Ds10n	**0.3**	0
Ds50n	**0.195**	0.078
Ds100n	**0.169**	0.084

Nevertheless, *ARTIVA* retains its superiority in F1-score for two of the three benchmark datasets (Table 1.3). The reason is that *TGS* can not overcome *ARTIVA* in rejecting incorrect edges, measured with a metric named 'precision' (Table 1.4). Precision is the ratio of the number of correct edges in a model to the total number of edges in the model; therefore, precision represents how good an algorithm is in rejecting incorrect edges (Liu et al., 2016). F1-score is the harmonic mean of recall and precision. For that reason, F1-score indicates how good an algorithm is in balancing recall and precision.

Table 1.3: The F1-scores of *TGS* and *ARTIVA* for the Benchmark Datasets. For each dataset, the highest F1-score is boldfaced.

Dataset	**TGS**	**ARTIVA**
Ds10n	**0.261**	0
Ds50n	0.069	**0.082**
Ds100n	0.057	**0.083**

Table 1.4: The Precision of *TGS* and *ARTIVA* for the Benchmark Datasets. For each dataset, the highest precision is boldfaced.

Dataset	**TGS**	**ARTIVA**
Ds10n	**0.231**	0
Ds50n	0.042	**0.086**
Ds100n	0.034	**0.081**

The aforementioned observations imply that, although *TGS* is able to supersede *ARTIVA* in recall, the differences are not sufficient to compensate for lower precisions; as a consequence, *TGS*'s F1-scores remain lower than that of *ARTIVA*. We address this limitation in the subsequent contribution.

Contribution 2: Balancing Recall and Precision

To address the limitation, we investigate the incorrect edges that *TGS* fails to reject. It is found that a large number of such edges represent indirect regulatory relationships; if gene A regulates gene B which in turn regulates gene C, then genes A and C are said to have an indirect regulatory relationship. Such a relationship must not be represented as an edge which indicates a direct relationship.

Therefore, a refinement strategy is added in the short-listing step; this strategy employs an algorithm called 'Algorithm for the Reconstruction of Accurate Cellular NEtworks' (in short, *ARACNE*) to prevent the candidate regulators that are highly likely to have indirect regulatory relationships with the concerned gene from being short-listed (Margolin et al., 2006). With the modified short-listing step, we propose our second algorithm named 'TGS-Plus' or 'TGS+' (Pyne et al., 2020). This algorithm demonstrates a significant improvement in precision over *TGS* at an acceptable cost in recall. As a result, *TGS+* provides the finest balance between recall and precision; inevitably, it supersedes *ARTIVA* in F1-score (Table 1.5).

Table 1.5: The F1-scores of *TGS+* and *ARTIVA* for the Benchmark Datasets. For each dataset, the highest F1-score is boldfaced.

Dataset	**TGS+**	**ARTIVA**
Ds10n	**0.429**	0
Ds50n	0.066	**0.082**
Ds100n	**0.104**	0.083

In theory, the addition of the refinement strategy slightly increases the time complexity. However, in practice, the strategy results in shorter shortlists and thus saving more time in the final selection step. As a consequence, *TGS+* outpaces even *TGS* in runtime for all datasets (Table 1.6).

Table 1.6: The Runtime of *TGS+*, *TGS* and *ARTIVA* for the Benchmark Datasets. For each dataset, the fastest runtime is boldfaced.

Dataset	**TGS+**	**TGS**	**ARTIVA**
Ds10n	**5.515 s**	5.789 s	10 m 20 s
Ds50n	**22.034 s**	7 m 36 s	4 h 30 m 15 s
Ds100n	**1 m 4 s**	17 m 49 s	31 h 52 m 54 s

Nevertheless, we observe that the memory requirement of the final selection step keeps increasing at an exponential rate with the number of candidate regulators in the shortlist. This is alarming since shortlists may contain hundreds of genes for large-scale datasets having thousands of genes. Hence, we address this concern in the following contribution.

Contribution 3: Improving Memory-efficiency

To address the memory-inefficiencies, we inspect the final selection step. The root cause is found in the memory management scheme of the algorithm employed for conducting Bayesian network tests. This algorithm is known as *Bene* (Silander and Myllymäki, 2006). To select the final list of regulators for a concerned gene, *Bene* takes the shortlist of candidate regulators as input. Then, it enumerates all subsets of the short-listed candidate regulators. Consequently, each subset is scored with a scoring function. In the end, the subset with the highest score is selected as the final list. The memory-inefficiency arises because all the aforementioned subsets are kept together in memory during their score calculations. Hence, the memory requirement becomes proportional to the number of subsets, which is exponential to the number of short-listed candidate regulators.

We propose a novel memory management scheme based on the observation that calculating the score of one subset is independent of that of another subset. In this scheme, all subsets are not generated at once. Instead, a computer programme is used to take a subset as input, generate the next subset and replace the former subset with the latter. Therefore, the process starts with only the first subset in memory. As soon as its score is calculated, it is replaced with the second subset by the programme. The process iterates until the score of the last subset has been calculated. All through the process, a separate block of memory is allocated to store the highest-scoring subset.

After every iteration, the block reflects the highest-scoring subset among the subsets scored so far. As a result, after the last iteration, the block reflects the highest-scoring subset among all subsets.

The aforementioned scheme requires only two subsets – the current one and the hitherto highest-scoring one – to be stored in memory, along with the subset-generation programme. Therefore, the memory requirement becomes linear to the number of short-listed candidate regulators. At the same time, it does not require any additional time; we prove that this scheme has the same time complexity as that of *Bene*.

Hence, we replace *Bene* with the proposed scheme in *TGS*. It gives birth to our third algorithm that we named 'TGS - which is Light on memory', in short, *TGS-Lite* (Pyne and Anand, 2019a). This algorithm outputs the same networks as that of *TGS* since both of them use the same scoring function. Moreover, it has the same time complexity as that of *TGS*. The only difference is that the memory usage of *TGS* grows exponentially with the number of short-listed candidate regulators, whereas that of *TGS-Lite* grows only linearly (Figure 1.1).

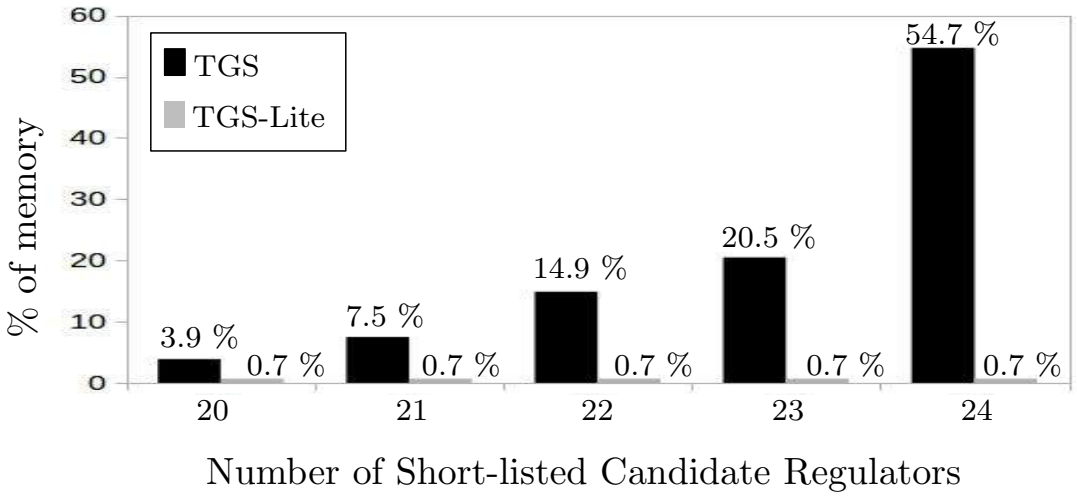


Figure 1.1: Differences in Memory Usage between *TGS* and *TGS-Lite*. The values represent percentages of memory used, where the total memory is 32 GB in size.

Similarly, we replace *Bene* with the proposed scheme in *TGS+*. The resultant algorithm is named *TGS-Lite+* (Pyne and Anand, 2019a). This algorithm reconstructs the same networks as *TGS+* within the same time complexity, yet, requires significantly less amount of memory.

Nonetheless, a major limitation is detected in all the algorithms we proposed so far. The limitation is that they tend to fail in capturing a particular type of edges known as 'transient edges'. These edges remain active for short periods of time but may trigger long-lasting cascading effects. Therefore, capturing transient edges is crucial for understanding the underlying gene-regulation process. We address this limitation in the following contribution.

Contribution 4: Capturing Transient Edges

We investigate the previously proposed algorithms to understand why they tend to miss transient edges. The bottleneck is found in the short-listing step. This step prepares a single time-invariant shortlist of candidate regulators for the concerned gene. The shortlist is then passed on to the final selection step, which may select different subsets of the shortlist for different time intervals. Therefore, the final lists may be different in distinct time intervals. However, all of them have to be subsets of the same shortlist.

The time-invariant shortlist is prepared based on the whole time-series dataset. As a result, the candidate regulators that share high mutual information with the concerned gene, across all time intervals, are short-listed. On the other hand, the candidate regulators that share high mutual information with the concerned gene, for a small number of time intervals, are less likely to be short-listed. However, such candidate regulators are most likely to be transient regulators of the concerned gene during those specific time intervals.

To capture the transient regulators, we design a novel short-listing step that generates one shortlist for every time interval. The shortlist specific to a time interval contains the candidate regulators that share statistically significant mutual information with the concerned gene, during that specific time interval. A novel algorithm is developed by replacing the short-listing step of *TGS* with the new one; we named this algorithm 'TGS - having Time-varying shortlists' or *TGS-T* (Pyne and Anand, 2019b). This algorithm captures considerably higher numbers of correct edges than that of *TGS*. As a consequence, *TGS-T* outperforms *TGS* in recall for two of the three benchmark datasets (Table 1.7).

Table 1.7: The Recalls of *TGS-T* and *TGS* for the Benchmark Datasets. For each dataset, the highest recall is boldfaced.

Dataset	**TGS-T**	**TGS**
Ds10n	0	**0.3**
Ds50n	**0.39**	0.195
Ds100n	**0.295**	0.169

On the other hand, *TGS-T*'s precisions, though competitive to that of *TGS*, remain consistently lower than the precisions of *TGS+*. To mitigate this issue, we attempt to incorporate the *ARACNE*-based refinement strategy used in *TGS+* into the short-listing step of *TGS-T*. However, it poses a challenge as *ARACNE* is incompatible with time-varying shortlists. We overcome this challenge by designing a novel refinement strategy that is compatible with time-varying shortlists. This strategy is named *ARACNE-T*. It utilises the theorem underlying *ARACNE* known as Data Processing Inequality (Margolin et al., 2006) and applies it on time-varying shortlists. By incorporating *ARACNE-T* into TGS-T, we develop a new algorithm named *TGS-T+* (Pyne and Anand, 2019b). As expected, *TGS-T+* obtains monotonically higher precisions than that of *TGS-T*. At the same time, the former retains the same recalls as that of the latter.

In terms of memory management, both *TGS-T* and *TGS-T+* inherit the inefficiencies from *TGS* and *TGS+*. To alleviate this issue, we propose two more algorithms by replacing their final selection steps with that of *TGS-Lite*; the resultant algorithms are named *TGS-T-Lite* and *TGS-T-Lite+*, respectively (Pyne and Anand, 2019b). These algorithms offer the same correctnesses at the same time complexities, yet, with significantly less memory.

Nevertheless, there remain opportunities for further improvements. We discuss some of those opportunities in the last chapter of this thesis.

1.2 Organisation of the Thesis

The thesis is organised in nine chapters. The chapters are briefly described below:

- **Chapter 1: Introduction.** This chapter provides an overview of the thesis.
- **Chapter 2: Motivation.** In this chapter, we discuss what motivated us to explore computational algorithms that serve the needs of biology and medicine.
- **Chapter 3: Literature Survey.** A critical review of such algorithms are conducted to discern their limitations. We find a major limitation in the computational efficiencies of the algorithms which reconstruct time-varying gene regulatory networks from time-series gene expression datasets.
- **Chapter 4: Problem Formulation.** The objective of this thesis is set to develop novel algorithms that can offer competitive correctness to that of the existing algorithms, yet, in a significantly more efficient manner.
- **Chapter 5: Improving Time-efficiency.** We develop our first algorithm. This algorithm outpaces the existing algorithms in runtime; moreover, it outperforms the latter in recall. However, the proposed algorithm is observed to suffer from poor precisions and exponential memory requirements.
- **Chapter 6: Balancing Recall and Precision.** To overcome poor precisions, we develop our second algorithm. This algorithm offers competitive precisions to that of the existing algorithms while being as time-efficient and recall-powerful as the previously proposed algorithm. Therefore, the only issue that remains is that of exponential memory requirements.
- **Chapter 7: Improving Memory-efficiency.** We resolve the issue by developing two novel algorithms. These algorithms are equivalent to the previously proposed algorithms, except the former algorithms have linear memory requirements. At this point, we reach the objective of the thesis, which was to develop algorithms that offer competitive correctness to that of the existing algorithms, in a more efficient manner. However, it is observed that the newly developed algorithms are unable to capture a particular type of edges known as transient edges; such edges remain active for a short period of time but may have a long-lasting effect. We choose to resolve this issue since the broader objective of the thesis is to advance the state-of-the-art of reconstruction algorithms.
- **Chapter 8: Capturing Transient Edges.** The aforementioned issue is mitigated by developing four more algorithms. These algorithms are equivalent to the four algorithms proposed previously, except the former algorithms are able to capture significantly higher numbers of edges.
- **Chapter 9: Conclusions and Future Directions.** Finally, we summarise the contributions and discuss few future directions.

At the end of every chapter, a section called 'Chapter Summary' is added. It presents a bird's eye view of how that chapter relates to the chapters preceding it. Naturally, this rule excludes chapters 'Introduction' and 'Conclusions and Future Directions'.

Chapter 2

Motivation

2.1 Important Questions

Study of biological systems finds its origin at the dawn of human consciousness. Generations of researchers have been dedicating their lives in pursuit of two important answers. Firstly, the big philosophical question of "What is life?" How does a living system behave the way a non-living system can not? The answer would not be complete unless we understand how a dynamic living system progresses through different stages of its life. However, living systems typically do not live alone. They interact with other living systems and environmental resources. This brings us to our second question which is of immediate concern for human lives. A human being is a very dynamic and open system. He or she interacts with a large number of other living systems and natural resources. Some of the interactions are essential to carry out his or her natural developmental processes. On the other hand, some of them are harmful to the expected developmental progression. Deviations from the expected trajectory can cause diseases. Identifying disease-causing interactions helps us to devise preventive and diagnostic strategies for such diseases. At the same time, finding out the interactions whose effect can potentially nullify that of the harmful interactions is crucial for developing therapeutic strategies. To summarise, the second question asks "How can we apply our biological knowledge in sustaining health?" The following statistics on burdens of wide-spread diseases in India could help us to fathom the compelling need for finding out the answer.

- Cervical cancer: Every 8 minutes, we lose 1 woman (NICPR, 2016, Statistics)
- TB: $(1/4)^{th}$ of the global incidents every year; 2.1 million citizens in 2013 (Central TB Division, 2015, Chapter 2, p. 22)
- HIV: 86,000 newly infected citizens reported in 2015 (NACO and NIMS, 2015, Figure 3, Estimated New HIV Infections in India, 1998–2015)

and the numbers keep growing ...

These statistics inevitably raise some questions. Such as -

Q. Complex diseases, like - cancers and diabetes, develop stage-by-stage at the molecular level and may take decades before clinical symptoms start appearing. By monitoring an apparently healthy individual at the molecular level, can we predict whether such a disease is under development?

Q. Given data collected from healthy individuals and those suffering from a particular disease, can we find out some novel features that can differentiate between two classes of individuals?

Q. If so many of us are infected, why are not all of us sick? Can we identify a set of features that can distinguish individuals more susceptible to a particular disease than the rest?

2.2 How to Answer

One scientific approach to find the answers is to study a large number of individuals across time and under different conditions. For each individual, we need to collect measurements of the variables from different levels of systemic regulations. Some of such variables are listed below:

- Innate Variables
  - Molecular variables: Genes and Proteins
  - Clinical variables: blood pressure, blood glucose, co-morbidities
  - Demographic variables: Age, gender, ethnicity
- Environmental variables: Life style, pathogen load

The measurements of these variables need to be analysed to discover their interdependencies. This discovery can lead to preventive and therapeutic strategies. For example, if the interdependencies between the molecular and clinical variables are learnt in the context of a cancer, then it can be predicted in advance whether an individual is likely to develop clinical symptoms of the cancer by monitoring his or her molecular variables. Detection of the cancer in an early stage followed by appropriate treatments may lead to a complete cure.

2.2.1 How to Answer: The Systems Biology Approach

"Every object that biology studies is a **system of systems**." $\sim$ Francois Jacob, recipient of the Nobel Prize in Medicine, 1965, for the hypothesis that control of enzyme levels in all cells occurs through regulation of transcription. (Trewavas, 2006)

Every biological system is a community of smaller interacting systems. Let us consider the human being for example. Human lives progress through the coordination of diverse societies. Every society is driven by distinct individuals and their interactions. Each unique individual gains its abilities by the proper functioning of multiple function-specific systems inside him or her, such as the nervous system. Every such system achieves its function with the help of multiple coordinating organs who take care of specific parts of the desired function. Each organ is organised into multiple layers of cross-talking tissues. Tissues, in turn, are collections of interacting cells. The cell is considered as the "basic structural, functional, and biological unit of all known living organisms" (Wikipedia, a). Human cells do not only communicate with each other but also with environmental stimuli and other living cells, like - bacteria and viruses.

Given an external or internal stimulus, the cell responds to it through an intricate mechanism. Unravelling this mechanism more and more precisely is the secret behind

understanding why different cellular systems behave differently under the same condition and why the same cell behaves differently under different conditions. The leading theory of how signals flow through a cell can be summarised as follows: The cellular system has multiple sensors or receptors; each assigned to sense a particular set of stimuli (Alon, 2006, Chapter 2). The receptors assigned to sense external stimuli are generally found on the cell membrane. On the other hand, the internal-stimuli receptors are located inside the cell. Given a stimulus, the corresponding receptors sense the input signal and compare it with a reference signal. Depending on whether the input varies significantly from the reference with respect to some threshold, the receptors initiate a chain of events, known as the **Signal transduction pathway** of the given stimulus. Typically, a signal transduction pathway involves a series of protein-protein interactions. It begins with the activation of a protein molecule specific to the receptor. This protein molecule, in turn, travels and activates another intermediate protein molecule. The process continues until the target protein molecule, known as the **Transcription Factor (TF)**, is activated. Once activated, the TF reaches to the DNA and activates the target gene. The activated gene produces a particular type of molecule called messenger RNA (mRNA) through a process named as **transcription**. Number of copies of mRNA transcribed from a particular gene per unit volume of the cellular environment at a specific time point is referred to as the **gene expression** of that gene at that time point. Transcribed mRNAs travel to ribosomes and get converted to corresponding Amino Acid chains; this conversion is called **translation**. Each amino acid chain folds into a 3D structure, forming a protein. These newly produced proteins again cause a cascade of protein-protein interactions to finally generate the cellular response against the aforementioned stimulus.

From the previous paragraph, it is perceived that the response of the cellular system emerges from a mechanistic interplay between different inter-dependent components of the system. Hence, if we can decipher the inter-dependencies between them, we would be able to understand and predict how the system would behave under a particular condition. Therefore, the question remains, how to decipher the inter-dependencies between the components of a given cellular system?

2.2.1.1 Deciphering Dependencies: Traditional Experimentation-only Approach

The traditional approach is to perturb a system variable of interest and study how changes in its values affect the values of other variables of interest (Markowetz and Spang, 2007). Typical perturbation strategies are to completely (knockout) or partially (knockdown) block the component corresponding to the system variable. For example, given a set of genes, the aim is to reverse engineer how they are dependent on each other. In that case, each gene can be perturbed by experimentally removing it from the DNA (knocked out) or reducing the number of mRNAs transcribed from it (knocked down); and then observing how it affects the expressions of the rest of the genes. In addition, multiple genes can be perturbed simultaneously to study their combinatorial effect on the rest of the genes.

The limitation of this approach is that the number of perturbation experiments grows exponentially with the number of system variables. Hence, when the number of system variables of interest is very large, the effort-time-cost required for the experimentation becomes prohibitive. For example, there are around $25,000$ genes in human DNA. Therefore, inferring their interdependencies through the experimentation-only approach is infeasible. In such a case, we need complementary approaches that can significantly

reduce the burden of experimentations.

2.2.1.2 Deciphering Dependencies: The Computational Systems Biology (CSB) Approach

The CSB approach is a complementary approach for reducing the burden of experimentations. This approach requires data collected by simultaneously measuring the system variables. Then, a computational model is constructed from the observed data. This model indicates the potential inter-dependencies among the variables. Therefore, perturbation experiments are performed only to verify potential inter-dependencies.

Graphical Models or Network Models In the CSB approach, a specific type of computational models, known as graphical or network models, is found to be very convenient for visualising inter-dependencies among a large number of variables (Markowetz and Spang, 2007; Raval and Ray, 2016). In a network model, the variables are represented as nodes and their dependency relationships are represented as edges. Absence of an edge between a pair of variables signifies their mutual independence. On the other hand, the presence of an edge implies that they are not mutually independent. The edge weight, if any, represents a quantitative measure of their dependence. Reverse engineering such a network from an input dataset is known as the **Network Reconstruction** task. The task can be formally defined as follows:

- The input is a **data matrix** of dimensions $(V \times N)$; it contains N measurements for each of the V variables of interest.
- The output is a **network adjacency matrix** of dimensions $(V \times V)$. The $(i, j)^{th}$ element in the matrix represents the dependency relationship between the i^{th} and the j^{th} variables. It can be noted that the **data structure** used to physically store the output network may not necessarily be an adjacency matrix. Depending upon the properties of the network and the desired operations, an efficient data structure can be chosen.

The computational algorithms designed to accomplish the aforementioned task is known as network-reconstruction algorithms (henceforth, simply **reconstruction algorithms**). Once the network is reconstructed, the **network-analysis** and **network-visualisation** techniques are applied on it. Mainly, the following two types of analyses are performed on the reconstructed networks:

- Firstly, statistical and functional analyses are performed to check whether the reconstructed network can explain the known functionalities, if any, of the concerned system. As an example, a network that models a system responsible for fast responses to external stresses must have a statistically significantly shorter mean-path-length than that of a system with a slower response. As another example, the genes known to be functionally involved in the development of butterfly's wings must have considerably more inter-connections in a network that models the metamorphosis stage than in a network that models another stage.
- Secondly, new experiments are designed to verify the edges for which prior knowledge is non-existent or limited. If the experimental results verify an edge, then the domain knowledge is enriched. Otherwise, the edge may be considered as incorrect. Consequently, the corresponding reconstruction algorithm can be modified to reject such incorrect edges.

Thus, computer-based reconstructions do not replace experiment-based approaches. Rather, the former attempt to accelerate the latter by narrowing down the experimental search space.

Focus of the Literature Survey The network reconstruction task is the scope of this thesis (Figure 2.1).

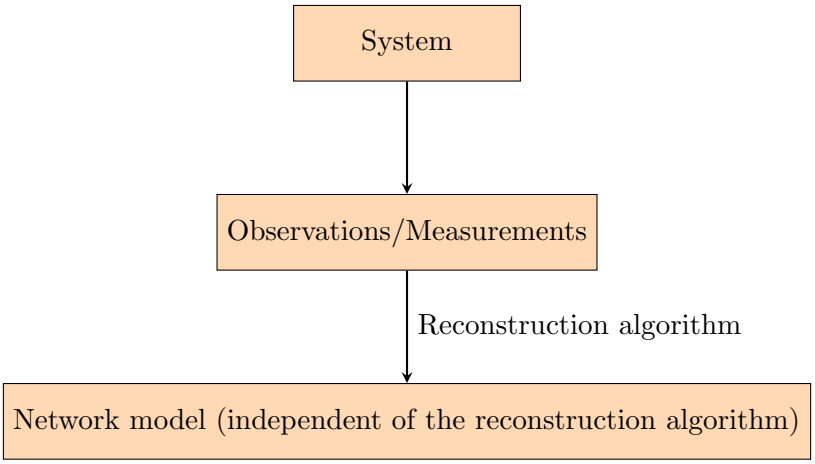


Figure 2.1: **Network Reconstruction.** For a system of interest, values of its variables are measured across time and under different conditions. These measurements are inputted into a reconstruction algorithm that outputs a network model of the concerned system. For each class of network models, multiple reconstruction algorithms can be designed to suit different types of measurements.

Different types of network models are designed to address the network reconstruction task. Moreover, for each type of models, a range of reconstruction algorithms is proposed. Hence, our literature survey is aimed at finding answers to the following questions:

Q. What are the different types of network models designed to represent inter-dependencies of variables in cellular systems?

Q. Given a particular type of network model, what are the reconstruction algorithms proposed to reconstruct it?

2.3 Chapter Summary

In this chapter, we discuss the motivation behind developing computational models to understand the progressions of developmental and disease-specific processes in dynamic biological systems, such as - human beings. In brief, the state of such a system can be defined as a function of the states of its variables. Various external and internal stimuli influence the states of the variables; thus, causing the system, as a whole, to transit into different states. In disease progression studies, identifying the key variables whose state changes are responsible for the transition from a healthy stage to a disease stage is crucial for prevention, early detection and treatment planning of the disease. In order to find out the key regulatory variables during a state transition, we need to reverse engineer or 'reconstruct' the inter-dependencies between the system variables at the states before and after the transition. Then, differential analyses need to be performed

to learn which variables are influenced to facilitate the system-wide state transition. Reconstructing inter-variable dependencies is traditionally done through wet-lab experiments. It involves changing the values of one variable at a time and analysing how it affects that of the other variables. However, when the number of variables is in thousands, the traditional approach becomes cost-and-effort prohibitive. This is where computational modelling comes in aid. It does not replace the experimental approach; rather, it attempts to accelerate experimentations by narrowing down the search space. For computational modelling, spatio-temporal measurements of the system variables are inputted into a computer programme, which uses a 'reconstruction algorithm' to learn state-specific inter-variable dependency structures and outputs them as computational models. Different types of computational models are designed for this purpose. Among them, network models are widely used for their ease with computational visualisations and analyses. For that reason, we chose the network models as the scope of this thesis. In network modelling, system variables are modelled as nodes and two nodes are connected by an edge (directed or undirected depending upon the model) if changes in their values, as observed in the input data, are not statistically independent. Theoretical results prescribe that, in such models, pairwise independence (un-connected node-pairs) is more reliable than dependency relationships (connected node-pairs). It implies that experimental validations need to be performed only for the connected node-pairs. Thus, reconstruction algorithms play an important role in reducing the search space of expensive experimentations. In the next chapter, we critically review different types of network models and corresponding reconstruction algorithms.

Chapter 3

Literature Survey

With each major technological and scientific contribution, our understanding improves. It leads to a generation of stronger hypotheses, based on which, more realistic models come forward. In this chapter, we explore the evolution of network models and corresponding reconstruction algorithms. The network models represent inter-dependencies between components of a biological system. The corresponding algorithms attempt to reconstruct these network models from spatio-temporal measurements of those system components.

3.1 Co-expression Based Models

The simplest network models are co-expression based models. They are based on the co-expression heuristic. The heuristic says that, if values of two system components are observed to be fluctuating at the same time points, then it may imply that they are involved in executing the same function (Eisen et al., 1998).

3.1.1 Input Data: Single system, Multiple time points

Given a system under observation, let us assume that input data $\mathbf{D}$ is a $(p \times N)$ matrix (Figure 3.1). It contains measurements of p system components across N samples. Different samples may be collected at distinct time points or from different tissues or under varied conditions. When each sample is collected at a distinct time point, then it forms a time-series data with N time points. Suppose that V denotes the set of system components. Each system component $v \in V$ is modelled as a random variable X_v. Also, all system components together are modelled as a random vector $\mathbf{X} = (X_1, \ldots, X_p)$. Therefore, dataset $\mathbf{D}$ contains N realizations of $\mathbf{X}$, denoted as $\left(x^{(1)}, \ldots, x^{(N)}\right)$. The objective is to generate a network (or graph) $G = (V, E)$ where the edge-set E represents pairwise co-expression relationships among the system components in V.

3.1.2 Correlation Networks

The oldest notion of co-expression is linear correlation, for example - Pearson correlation coefficient (Bansal et al., 2007). The pseudocode is given in Algorithm 1 with a graphical representation in Figure 3.2 . Zero correlation is a strong indicator of independence (Markowetz and Spang, 2007, Paragraph 'Comparison to correlation networks') i.e. if there exists no edge between a pair of system components $\{i, j\}$ then it is highly likely that they are mutually independent. Bickel (2005) introduces a very realistic 'time-lag' concept that says two components co-expressed without any time delay can not have a

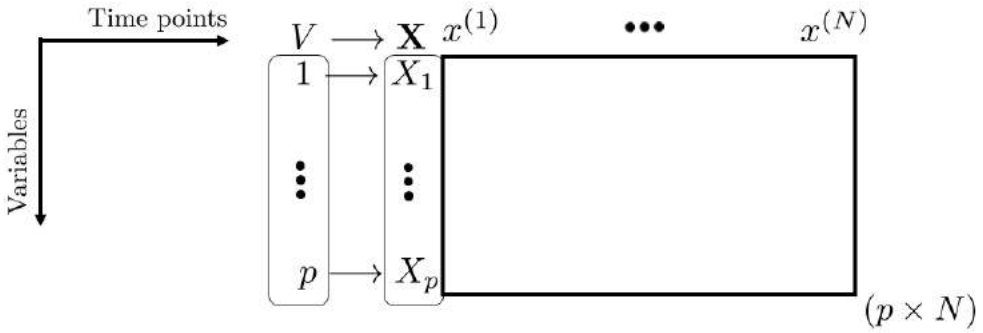


Figure 3.1: Input data matrix **D**. Given a system under observation, let us consider that **D** is a $(p \times N)$ matrix. It contains measurements of p system components across N samples. Different samples may be collected at distinct time points or from different tissues or under varied conditions. When each sample is collected at a distinct time point, it constitutes a time-series data with N time points. Suppose that V denotes the set of system components. Each system component $v \in V$ is modelled as a random variable X_v. Also, all system components together are modelled as a random vector $\mathbf{X} = (X_1, \ldots, X_p)$. Therefore, dataset **D** contains N realizations of $\mathbf{X}$, denoted as $\left(x^{(1)}, \ldots, x^{(N)}\right)$. An example of **D** can be a gene expression dataset having measurements of $p = 20,000$ genes across $N = 30$ time points from a single cell or averaged from a collection of cells.

regulator-regulatee relationship. Instead, it may imply that both of them are regulated by some third component.

Algorithm 1 Correlation Network Reconstruction

1: **Input:** Dataset **D**, vertex-set V, random vector $\mathbf{X}$.
2: **Output:** An undirected graph G.
3: Edge-set $E \leftarrow \varnothing$.
4: **for** each pair of random variables $\{X_i, X_j\} \in \mathbf{X}$ **do**
5: Compute r_{ij} = Pearson correlation coefficient of their expression vectors.
6: **if** $r_{ij} \geq$ predefined threshold **then**
7: $E \leftarrow E \cup (i, j)$. ▷ Add an undirected edge.
8: **end if**
9: **end for**

3.1.2.1 Merit(s) of Correlation Networks

- It is the simplest approach in terms of implementation and interpretation.
- An accurate reconstruction can be obtained even when $p \gg N$; this is a standard situation with high-dimensional biological dataset where number of samples is in order of tens and that of the variables is in order of ten thousands. For example, in a high-throughput human gene expression dataset, number of genes may be around 25 thousands whereas samples may be collected for only 30 time points.

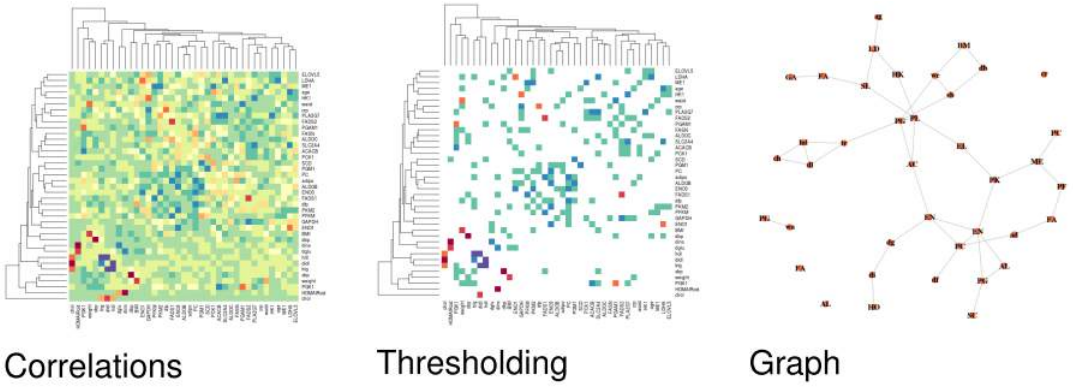


Figure 3.2: **Reconstruction of Correlation Network.** This figure is reproduced from Villa-Vialaneix 2014, slide 7. It provides a visual representation of how Algorithm 1 works. First, pairwise correlations are calculated and stored in a $(p \times p)$ matrix (the leftmost sub-figure), where $p =$ number of variables. The value at $(i, j)^{th}$ cell represents the Pearson correlation coefficient between the variables i and j. All the cell values are reduced to zero if it is less than a predefined threshold (sub-figure in the middle). Finally the matrix is visualized as an undirected graph (the rightmost sub-figure) where absence of edge between two nodes i and j implies that they are uncorrelated i.e. value at $(i, j)^{th}$ cell is zero. It can be noted that Pearson correlation coefficient is symmetric i.e. the values in the cells (i, j) and (j, i) of the correlation matrix should have the same value. But in this figure, the correlation matrix does not look exactly symmetric, which must be an unintentional mistake.

- Absence of an edge in the reconstructed correlation network is a strong indicator of mutual independence between its end points. Hence, it is very useful for reducing the edge search-space by removing mutually independent pairs.

3.1.2.2 Limitation(s) of Correlation Networks

- Correlation is a weak criterion for dependence. Presence of an edge between two system components i and j can not point out the exact dependency structure. There could be a range of dependency structures that can cause the correlation (Markowetz and Spang, 2007, Figure 1). Hence, given a correlation network, numerous perturbation experiments need to be performed to learn the exact dependency structure. Suppose that there is an edge between genes i and j. Then gene i can be perturbed to check how it affects the expression of gene j. In another experiment, the reverse strategy can be followed. However, for a dense correlation network with a large number of nodes, performing all possible perturbation experiments becomes infeasible. Thus, researchers attempt to devise more sophisticated models that can represent the underlying dependency structure more accurately.

3.1.3 Information Theoretic Models

3.1.3.1 Mutual Information Relevance Networks

Butte and Kohane (2000) propose to replace linear correlation coefficient with pairwise mutual information (Butte and Kohane, 2000, Equations 2 and 3) of two variables. The assumptions and the pseudocode (Algorithm 2) are given below. If mutual information

of two variables is zero then it implies that they are independent. Otherwise, the higher the mutual information, the less the probability of observing their co-expression by mere chance.

Assumptions This model makes the following two assumptions:

- In case of time-course data, observations made at any time point is independent of the observations at other time points. In practice, this assumption may hold if the sampling intervals are large enough.
- In case of a single time point but multiple samples data, observations in each sample is independent of the observations in the other samples.

Algorithm 2 Mutual Information Relevance Network Reconstruction

1: **Input:** Dataset $\mathbf{D}$, vertex-set V, random vector $\mathbf{X}$.
2: **Output:** An undirected graph G.
3: Edge-set $E \leftarrow \varnothing$.
4: **for** each pair of random variables $\{X_i, X_j\} \in \mathbf{X}$ **do**
5: Compute M_{ij} = Mutual information of their expression vectors.
6: **if** $M_{ij} \geq$ predefined threshold **then**
7: $E \leftarrow E \cup (i, j)$. ▷ Add an undirected edge.
8: **end if**
9: **end for**

Merit(s) of Mutual Information Relevance Networks

- Pearson correlation coefficient is a linear similarity measure; hence, can not model non-linear relationships. But pairwise mutual information can model such relationships.

Limitation(s) of Mutual Information Relevance Networks

- Non-zero mutual information is a weak criterion for dependence. So this model is prone towards reconstructing dense networks with a large number of false positive edges.

3.1.3.2 ARACNE

A more powerful information theoretic model is proposed by Margolin et al. to take care of the false positive edges (Basso et al., 2005; Margolin et al., 2006). First, it reconstructs a mutual information relevance network from the input dataset. Then it tries to identify potential false positive edges and prune them. The resultant algorithm is known as ARACNE (Algorithm 3). For identifying potential false positive edges, ARACNE utilises the converse of the Data Processing Inequality (DPI) principle.

Definition 1 (**Data Processing Inequality (DPI)**). *For any connected triplet* $\{i, j, k\}$, *if* i *and* k *do not have direct interaction, then* $M_{ik} \leq \min(M_{ij}, M_{jk})$.

Definition 2 (**Converse of Data Processing Inequality (DPI)**). *For any fully connected triplet (i.e. a triangle)* $\{i, j, k\}$, *if* $M_{ik} \leq \min(M_{ij}, M_{jk})$, *then* i *and* k *do not have direct interaction.*

Algorithm 3 ARACNE

Input: Dataset $\mathbf{D}$, vertex-set V, random vector $\mathbf{X}$.
Output: An undirected graph G.
Edge-set $E \leftarrow \varnothing$.

Reconstruct a mutual information relevance network
for each pair of random variables $\{X_i, X_j\} \in \mathbf{X}$ **do**
    Compute M_{ij} = Mutual information of their expression vectors.
    if $M_{ij} \geq$ predefined threshold **then**
        $E \leftarrow E \cup (i, j)$. ▷ Add an undirected edge.
    end if
end for
$G = (V, E)$.

Prune false positive edges
for each fully connected triplet (i.e. a triangle) $\{i, j, k\} \in G$ **do**
    if $M_{ik} \leq \min(M_{ij}, M_{jk})$ **then**
        $E \leftarrow E \setminus (i, k)$. ▷ Remove edge (i, k).
    end if
end for
return $G = (V, E)$.

Merit(s) of ARACNE

- ARACNE monotonically reduces the number of false positive edges in the mutual information relevance networks.

Limitation(s) of ARACNE

- It is assumed that the converse of the DPI principle holds, which is not necessarily true. Hence, ARACNE may prune true positive edges representing direct interactions. The authors of ARACNE admit this flaw as mentioned by (Bansal et al., 2007, Section 'ARACNE').

3.2 Conditional Independence (CI) Models

Distinguishing direct dependencies from the indirect ones remains a challenge with the co-expression based models. So the concept of conditional independence is introduced to address it.

Definition 3 (**Conditional Independence (CI)**). *Let us assume that X, Y are two random variables and Z is a set of random variables. They follow the joint probability distribution function $\mathcal{P}$. Then X is said to be conditionally independent of Y given Z (written as $X \perp Y \mid Z$) if and only if*

$$\mathcal{P}(X = x \mid Y = y, Z = z) = \mathcal{P}(X = x \mid Z = z) \tag{3.1}$$

Otherwise, it is said that X is conditionally not independent of Y given Z (written as $X \not\perp Y \mid Z$).

In CI models, two system components i and j are connected with an edge if and only if the corresponding random variables X_i and X_j are conditionally not independent given X_k, a subset of the rest of the random variables. Whether the edge is directed or undirected, that also depends on the model itself. Different classes of CI models are developed depending on how X_k is defined. They are -

- Full CI Models: $X_k = \mathbf{X} \setminus \{X_i, X_j\}$
- Low Order CI Models: $X_k \subset \mathbf{X} \setminus \{X_i, X_j\}$
- Bayesian Network: $X_k = \{X_S : \text{For all } X_S \subseteq (\mathbf{X} \setminus \{X_i, X_j\})\}$

These CI models are comparatively reviewed in the next sections.

3.2.1 Full Conditional Independence (CI) Models / Markov Random Fields (MRFs) / Markov Networks (MNs)

The Full CI models ask whether the observed correlation between two random variables can be explained by the rest of the variables or not. By definition, two system components i and j are connected with an undirected edge if and only if the corresponding random variables X_i and X_j are conditionally not independent given $X_{rest} = \mathbf{X} \setminus \{X_i, X_j\}$, the rest of the random variables in $\mathbf{X}$.

3.2.1.1 Gaussian Graphical Models (GGMs)

Full CI models are called GGMs when the given random vector $\mathbf{X}$ follows a multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$. Here μ is the mean vector of dimension p (i.e. p-vector) and $\Sigma = (\sigma_{ij})$ is the covariance matrix of dimension $(p \times p)$. Σ is a symmetric matrix since $(\sigma_{ij}) = (\sigma_{ji})$ = Covariance between the variables X_i and X_j. GGM assumes that Σ is invertible i.e. $K = \Sigma^{-1}$ exists. K is called the precision matrix or concentration matrix. Since inverse of an invertible symmetric matrix is also a symmetric matrix (Proof at Deriso (2013)), K is a symmetric matrix. The value $\rho_{ij} = \left(-k_{ij}/\sqrt{k_{ii}, k_{jj}}\right)$ is known as the partial correlation coefficient (pcc) between X_i and X_j, where k_{ij} stands for the $(i,j)^{th}$ entry in K. Then it satisfies the bi-implications for $i \neq j$ that

$$k_{ij} = 0 \iff \rho_{ij} = 0 \iff X_i \perp X_j \mid X_{rest} \tag{3.2}$$

An undirected edge is drawn between two system components i and j if and only if k_{ij} has a non-zero value i.e. the correlation between X_i and X_j can not be explained by the rest of the variables (Algorithm 4).

In practice, GGM faces a challenge when $p \gg N$, which is a very common scenario with high-throughput experiments. In such a case, Σ becomes non-invertible.

3.2.1. Theorem. *Σ is non-invertible when $p \gg N$.*

Proof.

$\Sigma = \frac{1}{N-1}\left(\mathbf{D} - \overline{\mathbf{D}}\right)^T \left(\mathbf{D} - \overline{\mathbf{D}}\right)$ (Schäfer and Strimmer, 2005, Section 'Graphical Gaussian models')

where $\mathbf{D} = (d_{ij})$ and $\overline{\mathbf{D}} = \left(\ \overline{d_{ij}}\ \right)$
s. t. $\overline{d_{i\cdot}} = \frac{1}{N}\sum_{j=1}^{N} d_{ij}$ i.e. the mean value of the random variable X_i.

Algorithm 4 Gaussian Graphical Model (GGM) Reconstruction

Input: Dataset $\mathbf{D}$, vertex-set V, random vector $\mathbf{X}$.
Output: An undirected graph G.
Edge-set $E \leftarrow \varnothing$.
Estimate the concentration matrix $K = \Sigma^{-1}$.
for each pair of random variables $\{X_i, X_j\} \in \mathbf{X}$ **do**
    if $k_{ij} \neq 0$ **then**
        ## $k_{ij} = (i,j)^{th}$ entry in K.
        $E \leftarrow E \cup (i,j)$. ▷ Add an undirected edge (i,j).
    end if
end for
return $G = (V, E)$.

Thus both $\mathbf{D}$ and $\overline{\mathbf{D}}$ matrices are of dimension $(p \times N)$. Hence $\left(\mathbf{D} - \overline{\mathbf{D}}\right)$ is a matrix of dimension $(p \times N)$. Therefore

$$\begin{aligned} \operatorname{rank}\left(\mathbf{D} - \overline{\mathbf{D}}\right) = \operatorname{rank}\left(\mathbf{D} - \overline{\mathbf{D}}\right)^T &\leq \min\left(p, N\right) \\ &\leq N \end{aligned}$$

Since for any two multipliable matrices A and B, $\operatorname{rank}(AB) = \min\left(\operatorname{rank}(A), \operatorname{rank}(B)\right)$,

$$\begin{aligned} \operatorname{rank}(\Sigma) &= \operatorname{rank}\left(\left(\mathbf{D} - \overline{\mathbf{D}}\right) \cdot \left(\mathbf{D} - \overline{\mathbf{D}}\right)^T\right) \\ &= \min\left(\operatorname{rank}\left(\mathbf{D} - \overline{\mathbf{D}}\right), \operatorname{rank}\left(\mathbf{D} - \overline{\mathbf{D}}\right)^T\right) \\ &\leq N \\ &\ll p \qquad \implies \Sigma \text{ is non-invertible.} \quad \square \end{aligned}$$

In a seminal paper, Schäfer and Strimmer (2005) address this issue. Algorithm 5 is proposed and implemented as an R package named 'GeneTS'.

The major contribution of GeneTS is twofold:

- It replaces matrix inverse with Moore-Penrose pseudoinverse (Penrose, 1955) $\Sigma^+ = VD^{-1}U^T$, where V and U are eigenvectors of $\Sigma^T\Sigma$ and $\Sigma\Sigma^T$, respectively; D is a square diagonal matrix of rank $\leq \min(p, N)$ and so trivially invertible. Their values are obtained by Singular Value Decomposition (SVD) of Σ into UDV^T. This strategy eliminates the challenge of not being able to estimate GGM when $p \gg N$.
- The Bootstrap Aggregation (bagging) (Breiman, 1996) scheme is incorporated to reduce variance in the input data.

A computationally efficient algorithm for biological GGM inference is proposed in Wang et al. (2016) and implemented as an R package called 'FastGGM'. Through simulation, it demonstrates its ability to accurately (Area Under ROC Curve (AUROC) = 0.96) reconstruct GGM in a high-dimensional setting $\{N = 1,000, p = 10,000\}$ (Wang et al., 2016, Table 1). FastGGM makes the sparsity assumption that the maximum degree of the vertices in the true GGM is $o\left(\sqrt{N}/\log p\right)$. In addition to providing partial correlation coefficient between the random variables corresponding to two end points of an edge, it also calculates the p-value and confidence interval for the edge.

Algorithm 5 GeneTS GGM Reconstruction

Input: Dataset $\mathbf{D}$, vertex-set V, random vector $\mathbf{X}$, number of bootstrap samples to generate B.
Output: An undirected graph G.
Generate B number of bootstrap samples with replacement from $\mathbf{D}$. ▷ Bootstrap Aggregation (bagging).
for each bootstrap sample b **do**
    Estimate the concentration matrix $K_b = {\Sigma_b}^+$. ▷ Moore-Penrose pseudoinverse.
    Initialize adjacency matrix $\mathcal{A}_b$ of dimension $(p \times p)$ for the to-be-reconstructed graph with zeroes in each cell.
    for each pair of random variables $\{X_i, X_j\} \in \mathbf{X}$ **do**
        if $k_{ij} \neq 0$ **then**
            ## $k_{ij} = (i,j)^{th}$ entry in K_b.
            $\mathcal{A}_b(i,j)$ and $\mathcal{A}_b(j,i) \leftarrow k_{ij}$. ▷ Add an undirected edge (i,j).
        end if
    end for
end for
Final adjacency matrix $\mathcal{A} \leftarrow mean(\mathcal{A}_b)\ \forall b$.
return $\mathcal{A}$.

3.2.2 Low Order Conditional Independence (CI) Models

Let us discuss the motivation behind this class of models with the help of a practical example. Suppose a system component i simultaneously activates system components j and k. In such a case, Pearson correlation coefficient between the random variables X_j and X_k could be very high. But their partial correlation coefficient (pcc) given X_i is highly likely to be very low, indicating absence of any direct dependency. Therefore we can observe that a single third variable can be sufficient to identify an indirect dependency between a pair of variables. It significantly reduces the computational complexity of testing each pair w.r.t. all other variables, especially when they are large in number. In case of gene expression data, it is hypothesized that every gene is regulated by only a handful of other genes. If that is the case then a small subset of genes should be able to explain the indirect correlation, if any, between a given pair of genes. This insight motivates the researchers to come up with the First Order CI model (Magwene and Kim, 2004; De La Fuente et al., 2004; Wille and Bühlmann, 2006; Wille et al., 2004) for GGM where an undirected edge is drawn between two system components i and j if and only if $(\forall k \in \mathbf{X} \setminus \{i,j\})\,(X_i \not\perp X_j \mid X_k)$. This model automatically eliminates $p \gg N$ problem. Because for each test, instead of considering the whole $(p \times N)$-dimensional data matrix, only a sub-matrix of dimension $(3 \times N)$ is adequate for reconstruction. It results in an exact estimation of the GGM when $N \geq 3$. Similarly, the n^{th} Order CI model asks whether the observed correlation, if any, between two variables can be explained by any n sized subset of rest of the variables. Given a dataset, selection of the value of n depends upon the biological insight, computational constraints, etc.

3.2.3 Bayesian Network (BayesNet)

Pearl (1985) introduces BayesNet to generalize the idea of Conditional Independence (CI) models. Unlike, other CI models, it produces a Directed Acyclic Graph (DAG). A directed edge is drawn from the system component i to another component j if and only if their correlation can not be explained by any subset of rest of the variables. It is formally written as $X_i \not\perp X_j \mid X_S$ for all $X_S \subseteq \mathbf{X} \setminus \{X_i, X_j\}$. Thus BayesNet envelops

correlation networks and all the previously discussed CI models:

- Correlation networks when $X_S \leftarrow \varnothing$.
- Full CI models when $X_S \leftarrow \mathbf{X} \setminus \{X_i, X_j\}$.
- First Order CI models when $X_S \leftarrow X_k$ for all $X_k \in \mathbf{X} \setminus \{X_i, X_j\}$.

3.2.3.1 Merit(s) of BayesNet

- Due to testing w.r.t. all orders of Conditional Independence (CI), this model is guaranteed to produce the lowest number of edges among all CI models.
- The edge direction implies possible regulator-regulatee relationship.
- Each random variable X_v is described by a Local Probability Distribution (LPD) $\mathsf{p}(X_v)$. A key property of BayesNet is that the Joint Probability Distribution (JPD) $\mathsf{P}(\cdot)$ over all variables can be factorized as follows:

$$\mathsf{P}(\mathbf{X}) = \prod_{v \in \mathbf{V}} \mathsf{p}\left(X_v \mid \mathbf{X}_{\mathrm{pa}(\mathbf{v})}\right) \tag{3.3}$$

  where $\mathrm{pa}(v)$ is the set of all vertices from which there is an edge to vertex v; $\mathbf{X}_{\mathbf{pa}(\mathbf{v})}$ is the set of all random variables corresponding to the vertex set $\mathrm{pa}(v)$. With the help of this property, the variables can be partitioned into families. Efficient network estimation and analysis techniques are devised based on divide-and-conquer paradigm to utilize this property, especially when the network is very large (Nikolova et al., 2013).

3.2.3.2 Limitation(s) of BayesNet

- For each ordered pair of random variables (X_i, X_j), number of tests to be performed $= |X_S| = 2^{|\mathbf{X} \setminus \{X_i, X_j\}|} = 2^{(p-2)}$. And there are $2 \cdot \binom{p}{2}$ such ordered pairs possible. Hence, a total of $2 \cdot \binom{p}{2} \cdot 2^{(p-2)}$ tests are needed to reconstruct a BayesNet. Therefore, the computational complexity increases exponentially with the number of observed variables, rendering inference infeasible in high-dimensional setting. As discussed in (Markowetz and Spang, 2007, Section 'Score based structure learning'), the following strategy is generally adapted to overcome this issue:
  - Define a search space of possible models: if possible reduce the search space by applying prior knowledge;
  - Define a scoring function to compute fitness of each model in the search space with the given data
  - Construct an optimization problem to find out the fittest model
  - Solve the optimization problem
- BayesNet strictly prohibits cyclic dependencies. This is an unrealistic constraint since dynamic biological systems are known to possess feedback loops that facilitate adaptivity, which is one of the salient features of dynamic living systems.
- The biggest theoretical challenge with BayesNet is Markov equivalence. Two different BayesNets are called Markov equivalent if they possess the same underlying undirected graph and directed collider sub-graph(s), like - $X_i \rightarrow X_j \leftarrow X_k$. Two

such BayesNets are statistically indistinguishable. It implies that given a dataset, we can only infer to which Markov equivalence class the output BayesNet is likely to belong. But we can not point out the exact member of the class that represents the true dependency structure even when $N \longrightarrow \infty$.

3.3 Joint Network Inference (JNI) Models

In this section, the models under investigation help the researchers to answer how inter-dependencies between the components of a given system readjusts differently under different conditions so that the system, as a whole, can adapt to the changed condition.

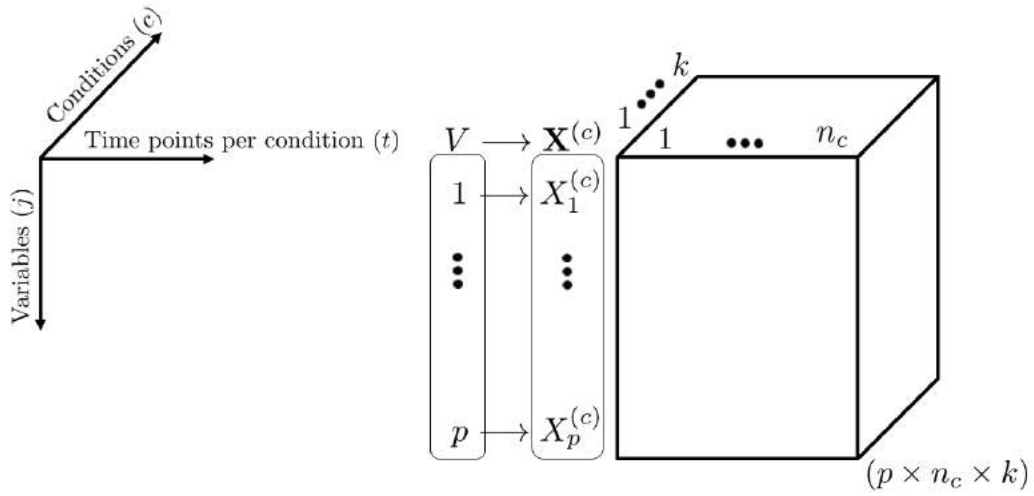


Figure 3.3: Input Data Tensor **D**: Given a system under observation, let us assume that input data **D** is a $(p \times n_c \times k)$ tensor. It contains measurements of p system components under k different conditions. For each condition c, there are n_c independent observations, suppose at n_c different time points. Total number of observations is $\sum_{c=1}^{k} n_c = N$. Each system component $v \in V$ is modelled as k random variables $\{X_v^{(c)} : \text{For all } c \in [k]\}$ corresponding to k different conditions; $[n]$ stands for the set of first n natural numbers starting from 1. Then all the system components can be modelled as $(p \times k)$ random variables $\{X_j^{(c)} : j \in [p],\ c \in [k]\}$. Such a dataset is very common in interventional experiments. Given a system, such as a cancer patient, k tissue samples are collected from the tumour. Then each of them are treated (intervened) with a different drug. Post intervention, gene expression of the same $p = 20,000$ genes are measured across multiple time points under each of the conditions. Being able to infer the changes in genetic dependency structures of tumour cells in response to different types of treatment, may help the researchers to identify the best treatment strategy against the cancer. It is to note that different conditions may also refer to different tissues from where the data is collected. Such spatio-temporal dataset can be very helpful to understand the mechanism behind spread of a disease across different parts of the body.

3.3.1 Input Data: Single system, Multiple conditions, Multiple time points per condition

The current dataset has one more dimension, namely conditions, in addition to those of the dataset described in Section 3.1.1 . Given a system under observation, let us assume that input data $\mathbf{D}$ is a $(p \times n_c \times k)$ tensor [Figure 3.3]. It contains measurements of p system components under k different conditions. For each condition c, there are n_c independent observations, suppose at n_c different time points. Total number of observations is $\sum_{c=1}^{k} n_c = N$. Each system component $v \in V$ is modelled as k random variables $\{X_v^{(c)}\}$ corresponding to k different conditions i.e. for all $c \in [k]$; $[n]$ stands for the set of first n natural numbers starting from 1. Then all the system components can be modelled as $(p \times k)$ random variables $\{X_j^{(c)} : j \in [p]\,,\ c \in [k]\}$. The objective is to reconstruct k graphs that represent dependency structures among the system components in k different conditions.

3.3.2 Independent Network Inference (INI) with Gaussian Graphical Models (GGMs)

Let us assume that there are random variables $\left(X_j^{(c)} : \forall j \in [p]\right) \sim \mathcal{N}\left(0, \Sigma^{(c)}\right)$. One naive approach is to independently estimate the concentration matrix $K^{(c)}$ for each specific condition c . In Gaussian framework, its elements $k_{jj'}^{(c)}$ can be approximated by finding the linear regression coefficients $\beta_{jj'}^{(c)}$ in the $(k \times p)$ linear regression equations 3.4 . From the relationship between $k_{jj'}^{(c)}$ and $\beta_{jj'}^{(c)}$, as given in Equation 3.5 , it can be said that $\left(k_{jj'}^{(c)} = 0\right) \iff \left(\beta_{jj'}^{(c)} = 0\right) \iff \left(X_j \perp X_{j'} \mid X_{rest}\right)$.

$$X_j^{(c)} = \mathbf{X}_{\setminus j}^{(c)^T} \beta_j^{(c)} + \epsilon_j^{(c)} \qquad \forall j \in [p], \forall c \in [k] \tag{3.4}$$

where

$$\begin{aligned}
\mathbf{X}_{\setminus j}^{(c)^T} &= \left(X_{j'}^{(c)}\right) \qquad \forall j' \in [p] \setminus j \quad \triangleright \text{Random } (p-1)\text{-vector} \\
\beta_j^{(c)} &= \left(\beta_{jj'}^{(c)} : j' \in [p] \setminus j\right)^T \quad \triangleright \text{Linear regression coefficients. } (p-1) \text{ vector.} \\
K^{(c)} &= \left(k_{jj'}^{(c)}\right) \qquad (\forall j, j' \in [p]) \quad \triangleright \text{Elements of concentration matrix} \\
\beta_{jj'}^{(c)} &= -\frac{k_{jj'}^{(c)}}{k_{jj}^{(c)}} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (3.5) \\
\left(\epsilon_j^{(c)} : \forall j \in [p]\right) &\sim \mathcal{N}\left(0, \Sigma^{(c)}\right) \quad \triangleright \text{Gaussian centered error terms}
\end{aligned}$$

3.3.2.1 Limitation(s) of Independent Network Inference (INI)

- Independent estimation can not produce statistically acceptable model when number of observations in a condition is small or there are lots of missing/noisy data points. This is so because it reconstructs one network per condition solely depending on the data available for that condition.
- In line with the previous point, INI can not utilize prior knowledge shared among the condition-specific datasets. Suppose the data is collected across the following three conditions from a cancer patient:

– Malignant (diseased) tissue before any treatment

– The same tissue after chemotherapy

– Followed by a radiation therapy

In that case, a chain graph hierarchy exists between the conditions. A superior inference strategy can be developed that can utilise this knowledge for reconstructing more accurate models of each condition.

3.3.3 Joint Network Inference (JNI) with Gaussian Graphical Models (GGMs)

JNI approach is used to overcome the limitations of INI by utilising the relationships between different conditions under study [Figure 3.4]. As shown in (Villa-Vialaneix et al., 2014, Equation (2)), all condition-specific concentration matrices can be jointly estimated by maximizing the pseudo-loglikelihood (Wikipedia, b) Equation 3.6 w.r.t. the concentration matrices.

$$\arg\max_{\mathbf{K}} \mathcal{L}\left(\mathbf{K} \mid \mathbf{X}\right) = \arg\max_{\mathbf{K}_j^c} \sum_{c=1}^{k} \sum_{j=1}^{p} \sum_{t=1}^{n_c} \log \mathsf{P}\left(\mathsf{X}_{\mathsf{tj}}^{\mathsf{c}} \mid \mathsf{X}_{\mathsf{t},\backslash\mathsf{j}}^{\mathsf{c}}, \mathbf{K}_{\mathsf{j}}^{\mathsf{c}}\right) \tag{3.6}$$

Efron (1981) shows that the dual of pseudo-loglikelihood Equation 3.6 is equivalent to jointly solving p equations 3.7 [(Villa-Vialaneix et al., 2014, Equation (3)).

$$\arg\min_{\beta_j} \left(\underbrace{\frac{1}{2}\beta_j^T \hat{\Sigma}_{\backslash j,\backslash j}\beta_j + \beta_j^T \hat{\Sigma}_{j,\backslash j}}_{\mathfrak{E}} \right) \qquad \forall j \in [p] \tag{3.7}$$

where

$$\beta_j = \left(\left(\beta_j^{(1)}\right)^T, \ldots, \left(\beta_j^{(k)}\right)^T\right)^T \in \mathbb{R}^{(k\cdot(p-1))} \equiv (k\cdot(p-1))\text{-vector}$$

$\Sigma =$ Covariance matrix

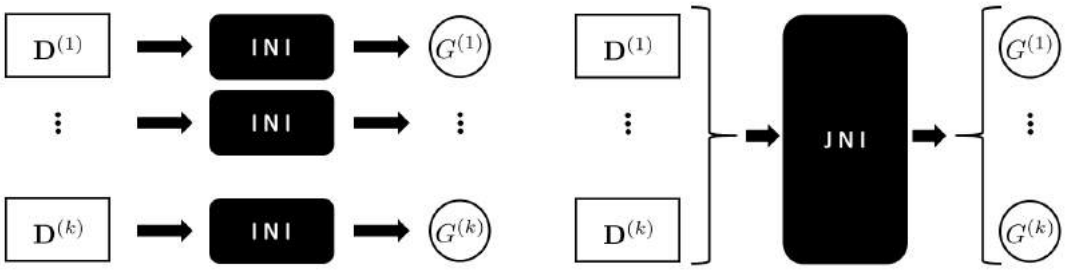
$$= \begin{array}{c} \\ X_1^{(1)} \\ \vdots \\ X_p^{(1)} \\ \vdots \\ X_1^{(k)} \\ \vdots \\ X_p^{(k)} \end{array} \begin{array}{c} \begin{array}{ccccccc} X_1^{(1)} & \cdots & X_p^{(1)} & \cdots & X_1^{(k)} & \cdots & X_p^{(k)} \end{array} \\ \begin{pmatrix} \cdot & \cdots & \cdot & \cdots & \cdot & \cdots & \cdot \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \cdot & \cdots & \cdot & \cdots & \cdot & \cdots & \cdot \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \cdot & \cdots & \cdot & \cdots & \cdot & \cdots & \cdot \\ \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\ \cdot & \cdots & \cdot & \cdots & \cdot & \cdots & \cdot \end{pmatrix}_{(kp\times kp)} \end{array}$$

$\hat{\Sigma} =$ Estimated value of $\Sigma \in \mathbb{R}^{(kp\times kp)}$

$\hat{\Sigma}_{\setminus j,\setminus j} = \hat{\Sigma}$ deprived of the rows and columns corresponding to $\{X_j^{(c)} : \forall c \in [k]\}$
$\in \mathbb{R}^{(k\cdot(p-1)\times k\cdot(p-1))}$

$\hat{\Sigma}_{j,\setminus j}^{(c)} = \hat{\Sigma}$ deprived of all the rows except $X_j^{(c)}$ for $c \in [k]$
$\in \mathbb{R}^{(p-1)} \equiv (p-1)$-vector

$$\hat{\Sigma}_{j,\setminus j} = \left(\left(\hat{\Sigma}_{j,\setminus j}^{(1)}\right)^T, \ldots, \left(\hat{\Sigma}_{j,\setminus j}^{(k)}\right)^T\right)^T$$
$\in \mathbb{R}^{(k\cdot(p-1))} \equiv (k\cdot(p-1))$-vector



(a) Independent Network Inference (INI) (b) Joint Network Inference (JNI)

Figure 3.4: **Difference between INI and JNI strategies.** The whole input dataset is denoted by $\mathbf{D}$. $\mathbf{D}^{(c)}$ represents the sub-dataset specific to the condition c. Similarly, $G^{(c)}$ stands for the reconstructed graphical model of condition c. INI uses only the sub-dataset $\mathbf{D}^{(c)}$ to reconstruct $G^{(c)}$. On the other hand, JNI takes the whole dataset $\mathbf{D}$ as input and reconstructs all $G^{(c)}$s simultaneously through information sharing.

It can be noted that the optimisation problem in Equation 3.7 is a Quadratic Programming Problem (QPP) (Wikipedia, c) of dimension $((p-1)\cdot k)$.
JNI of real biological networks is proved to be a challenging task. Some of the major challenges include:

Challenge I: Fostering Sparsity Biological regulatory networks are believed to be sparse i.e. only a handful of system components play the role of hubs and regulate a large number of other system components (Barabási and Albert, 1999; Barabasi and Oltvai, 2004). Therefore, JNI must be able to reconstruct sparse networks from high-dimensional data.

Challenge II: Preserving Commonality: JNI, by definition, must be able to preserve the condition-independent dependency relationships among the system components, while inferring the condition-specific dependencies. A motivational example can be found in 'The Diogenes project' (`http://www.diogenes-eu.org/`). In one of its studies, data is acquired from 204 obese women participants from different parts of Europe under two different conditions: before and after a low-calorie diet. From this data, two condition-specific networks are reconstructed. It is found that around 92% of their edges are common (Villa-Vialaneix et al., 2014, p. 57, Figure 7). An intuitive justification can lie in the modularity of the complex biological systems, like - human cells. Each module perform very specific tasks. So the observed adaptiveness of the whole system under a different condition may be the result of re-adjustment in a small number of modules. The rest of the interactions, like - the ones responsible for house-keeping activities, are highly likely to be invariant to conditions.

Challenge III: Joint Estimation of Multiple Related Systems: In medical research, cohort studies play a very important role. A group of participants with shared characteristics (e.g. same species, healthy or suffering from different stages of the same disease) is observed for a period of time under different conditions (like - undergoing different types of treatments). An efficient JNI technique must be developed that can simultaneously reconstruct multiple individual-specific networks.

In the following sections, we discuss the models and corresponding reconstruction algorithms that address the aforementioned JNI challenges.

3.3.3.1 Joint Network Inference (JNI) for Fostering Sparsity

To infer a sparse concentration matrix $\mathbf{K}$, Banerjee et al. (2008) propose to add an ℓ_1-regularised penalty to the objective function such a way that the denser the $\mathbf{K}$, the heavier the penalty is. Friedman et al. (2008) incorporate this idea with the regression analysis framework of Least absolute shrinkage and selection operator (Lasso) (Tibshirani, 1996) for inferring sparse GGMs; the inference technique is appropriately named as **Graphical Lasso (gLasso)**. Chiquet et al. (2011) improve upon it and propose **Graphical Intertwined Lasso (giLasso)**. Firstly, it uses the ℓ_1 sparsity penalty in the objective function (3.8). Hence, the denser the network, the higher the sum of the regression coefficients, and therefore, the higher the penalty. Thus, denser solutions are discouraged, resulting in favour of the sparser solutions.

Secondly, an intertwined estimation $\hat{\Sigma}^{(c)}$ of condition-specific covariance matrix is proposed (Equation 3.9). The estimation is a linear combination of the covariance matrix $\Sigma^{(c)}$ specific to the current condition c and the arithmetic mean $\overline{\Sigma}$ of the covariance matrices across all conditions.

$$\underset{\beta_j}{\arg\min}\left(\mathfrak{E}+\underbrace{\sum_{c=1}^{k}\left(\frac{1}{n_c}\left\|\beta^{(c)}\right\|_1\right)}_{\text{sparsity penalty}}\right)\qquad\forall j\in[p]\tag{3.8}$$

where

$\beta^{(c)}=\left(\beta_j^{(c)}:j\in[p]\right)\in\mathbb{R}^{(p-1)\cdot p}$

$\|\cdot\|_1=\ell_1$-norm of '$\cdot$'

$$\hat{\Sigma}^{(c)}=\alpha\cdot\Sigma^{(c)}+(1-\alpha)\cdot\overline{\Sigma}\tag{3.9}$$

where

α : is a regularization parameter; default value $\frac{1}{2}$

$\overline{\Sigma}=\frac{1}{N}\sum_{c=1}^{k}\left(n_c\cdot\Sigma^{(c)}\right)$

In the same paper, Chiquet et al. (2011) propose another inference method, namely **Graphical cooperative-Lasso (gCoopLasso)**. It is inspired by Group Lasso (grpLasso) (Yuan and Lin, 2006), a special case of Lasso.

In gCoopLasso, covariates (predictor variables) are partitioned into groups while performing regression analysis of a given response variable. Only one group of covariates can be chosen for the final solution. Instead of grouping the covariates, gCoopLasso, groups pairwise interactions.

The idea is motivated by the observation in biological systems that, if a system component j' is found to be up-regulating another component j in one condition (denoted by $c1$), then it is vary rare that the same regulator j' would be found down-regulating the same regulatee j in another condition (denoted by $c2$).

In terms of regression analysis, the objective function (3.10) is more heavily penalised if $\beta_{jj'}^{(c1)}$ and $\beta_{jj'}^{(c2)}$ have different signs for any two conditions $c1,c2\in[k]$ than when they have the same sign.

$$\underset{\beta_j}{\arg\min}\left(\mathfrak{E}+\underbrace{\sum_{j'\in[p]\setminus j}\left(\left\|\left(\beta_{jj'}^{[1:k]}\right)_+\right\|_2+\left\|\left(-\beta_{jj'}^{[1:k]}\right)_+\right\|_2\right)}_{\text{sparsity penalty}}\right)\qquad\forall j\in[p]\tag{3.10}$$

where

$\beta_{jj'}^{[1:k]}=\left(\beta_{jj'}^{(1)},\ldots,\beta_{jj'}^{(k)}\right)^T$

$(\cdot)_+=\max(\cdot,0)$; Note: If input is a vector, then to be applied on each element.

Let us illustrate the aforementioned penalty function with an example. Given a response variable j, the regression coefficients of a predictor variable j' w.r.t. two

different conditions $\{c1, c2\}$ are $(a, b)^T$; the regression coefficients of another predictor variable j'' w.r.t. two different conditions are $(a, -b)^T$. a, b are positive real numbers.

$$\begin{aligned}
\text{Penalty for } j' \text{ is} &= \left(\left\| \begin{pmatrix} a \\ b \end{pmatrix}_+ \right\|_2 + \left\| - \begin{pmatrix} a \\ b \end{pmatrix}_+ \right\|_2 \right) \\
&= \left(\left\| \begin{pmatrix} a \\ b \end{pmatrix}_+ \right\|_2 + \left\| \begin{pmatrix} -a \\ -b \end{pmatrix}_+ \right\|_2 \right) \\
&= \left(\left\| \begin{pmatrix} a \\ b \end{pmatrix} \right\|_2 + \left\| \begin{pmatrix} 0 \\ 0 \end{pmatrix} \right\|_2 \right) \\
&= \left(\sqrt{a^2 + b^2} \right) \qquad (3.11)
\end{aligned}$$

$$\begin{aligned}
\text{Penalty for } j'' \text{ is} &= \left(\left\| \begin{pmatrix} a \\ -b \end{pmatrix}_+ \right\|_2 + \left\| - \begin{pmatrix} a \\ -b \end{pmatrix}_+ \right\|_2 \right) \\
&= \left(\left\| \begin{pmatrix} a \\ -b \end{pmatrix}_+ \right\|_2 + \left\| \begin{pmatrix} -a \\ b \end{pmatrix}_+ \right\|_2 \right) \\
&= \left(\left\| \begin{pmatrix} a \\ 0 \end{pmatrix} \right\|_2 + \left\| \begin{pmatrix} 0 \\ b \end{pmatrix} \right\|_2 \right) \\
&= (a + b) \qquad (3.12)
\end{aligned}$$

Since $(a + b)$ is strictly greater than $\sqrt{a^2 + b^2}$ for any positive real values of $\{a, b\}$, the penalty function imposes heavier penalty on edge $\{j'', j\}$ than on $\{j', j\}$, as expected. It can be observed that gCoopLasso replaces ℓ_1-regularisation with ℓ_2-regularisation. A comparative study of these two types of regularisation is discussed in Ng (2004); log0 (2013); Narayan (2014).

In the next section, it is reviewed how JNI methods attempt to address the underlying commonality between different condition-specific networks. While doing so, we encounter more sophisticated techniques that can simultaneously take care of sparsity and commonality requirements.

3.3.3.2 Joint Network Inference (JNI) for Preserving Commonality

In order to enforce similarity between condition-specific covariance matrices, Mohan et al. (2012) propose **perturbed-node joint graphical Lasso (pnjgLasso)**. The intuition behind this method is that most of the edges in the condition-specific networks are common; the difference between responses of the whole network under distinct conditions arises from perturbation of some of the nodes that in turn cause re-orientation of the edges stemming from those nodes. Therefore to preserve the common edges, a commonality penalty is added to the objective function [3.13].

$$\underset{\beta_j}{\arg\min} \left(\mathfrak{E} + \underbrace{\sum_{\substack{c,c' \in [k] \\ c \neq c'}} \sum_{j' \in [p] \setminus j} \left(\left\| \Sigma^{(c)}_{j,j'} - \Sigma^{(c')}_{j,j'} \right\|_2 \right)}_{\text{commonality penalty}} \right) \qquad \forall j \in [p] \qquad (3.13)$$

Danaher et al. (2014) propose to replace the ℓ_2-norm in the pnjgLasso commonality penalty with ℓ_1-norm (3.14). This method, known as **joint graphical Lasso (jgLasso)** outperforms the then competing methods on simulated and real lung cancer gene expression datasets.

$$\underset{\beta_j}{\arg\min} \left(\mathfrak{E} + \underbrace{\sum_{\substack{c,c' \in [k] \\ c \neq c'}} \sum_{j' \in [p] \setminus j} \left(\left\| \Sigma^{(c)}_{j,j'} - \Sigma^{(c')}_{j,j'} \right\|_1 \right)}_{\text{commonality penalty}} \right) \qquad \forall j \in [p] \tag{3.14}$$

Villa-Vialaneix et al. (2014) propose **consensus Lasso (cLasso)** to combine sparsity and commonality penalties in a single objective function 3.15.

$$\underset{\beta_j}{\arg\min} \left(\mathfrak{E} + \underbrace{\lambda \left\| \beta_j \right\|_1}_{\text{sparsity penalty}} + \underbrace{\mu \sum_c \left(w_c \left\| \beta^{(c)}_j - \beta^{(cons)}_j \right\|^2_2 \right)}_{\text{commonality penalty}} \right) \qquad \forall j \in [p] \tag{3.15}$$

where

$\{\lambda, \mu\}$: Regularisation parameters

w_c = Relative weight of condition c w.r.t. other conditions $\in \mathbb{R}$;

by default every condition can be assigned the same weight; like 1 or $\frac{1}{\sqrt{n_c}}$.

$\beta_j = \left(\left(\beta^{(1)}_j \right)^T, \ldots, \left(\beta^{(k)}_j \right)^T \right)^T \in \mathbb{R}^{(k \cdot (p-1))} \equiv (k \cdot (p-1))$-vector

$\beta^{(cons)}_j$: Regression coefficient values of the consensus network.

Every condition-specific network is encouraged to be similar to it.

$= \sum_c \left(\frac{n_c}{N} \cdot \beta^{(c)}_j \right)$

or a user-defined prior network can be used as the consensus network.

The performance of cLasso is comparatively examined with the aforementioned JNI methods in (Villa-Vialaneix et al., 2014, Section '4.1.3. Performance Comparisons') using the following simulation procedure:

- A well-recognized artificial scale-free network (titled the 'parent network') with 100 nodes (each node corresponds to a gene) and 200 edges is downloaded (Villa-Vialaneix et al., 2014, Section '4.1. Simulated Data').
- Cycles are removed from the parent network.
- k distinct children networks are produced by differently rewiring r% edges of the parent network; here, k and r are experimental parameters. Each of the children networks is considered to be representing a particular condition.

- Time-series gene expression data is generated from each child network using a random Gaussian process.

- The previously discussed methods are applied on this artificial gene expression dataset to reverse engineer the condition-specific children networks as GGMs. Their performance is measured in F1-score.

It is found that cLasso performs better than the other approaches for different values of k and r (Villa-Vialaneix et al., 2014, Tables 2 and 3). It achieves its highest F1-score, which is 0.86, with $k = 2$, $r = 5$, $(\forall condition\ c \in [k])\,(sample\ size\ n_c = 30)$, bootstrap resampling and utilising the parent network as the consensus network. However, Villa-Vialaneix et al. mention that this method could not address the situation when there are different numbers of samples (or time points) available for different conditions. This limitation opens up an immediate research opportunity.

So far, in this chapter, the JNI algorithms that deal with challenges I and II and produce GGMs as output are reviewed. But challenge III demands more powerful output models to capture such rich datasets. In the next section, we discuss the challenge III scenario, related models and corresponding reconstruction strategies.

3.3.4 Joint Network Inference (JNI) for Joint Estimation of Multiple Related Systems

In this section, the methods under investigation attempt to answer how different systems respond differently under the same condition. More precisely, how inter-dependencies between the components of a given set of systems re-adjust differently in response to the same stimulus, causing differential responses.

3.3.4.1 Input Data: Multiple related systems, Multiple conditions, Multiple time points per condition

The current dataset adds one more dimension, namely systems, in addition to those of the dataset described in Section 3.3.1 . Given I number of related biological systems under observation, let us assume that input data $\mathbf{D}$ is a $(p \times n_{c;i} \times k \times I)$ tensor [Figure 3.5]. Subset of the data corresponding to a particular individual system i is denoted by $\mathbf{D}_i$. The relationship between the given systems would be defined on case-by-case basis during the detailed discussion of the models. Each $\mathbf{D}_i$ contains measurements of p system components under k different conditions. For each condition c, there are $n_{c;i}$ independent observations, suppose at $n_{c;i}$ different time points. Total number of observations in $\mathbf{D}_i$ is $\sum_{c=1}^{k} n_{c;i} = N_i$. Similarly, the total number of observations in $\mathbf{D}$ is $\sum_{i=1}^{I} N_i = N$. The system components $v \in V$ are modelled as k random variables $\{X_v^{(c)}\}$ corresponding to k different conditions i.e. for all $c \in [k]$; $[n]$ stands for the set of first n natural numbers starting from 1. Then for each of the systems, all its components can be modelled as $(p \times k)$ random variables $\{X_j^{(c)} : j \in [p]\,,\ c \in [k]\}$. The objective is to reconstruct system-specific networks. Number of networks to be reconstructed for each system depends upon the methods. Some methods reconstruct a single network for each system, known as a summary network. On the other hand, there are methods that can model temporal progression of the system-specific dependency structure by reconstructing multiple time-varying networks for each individual system. These methods are reviewed in the following sections.

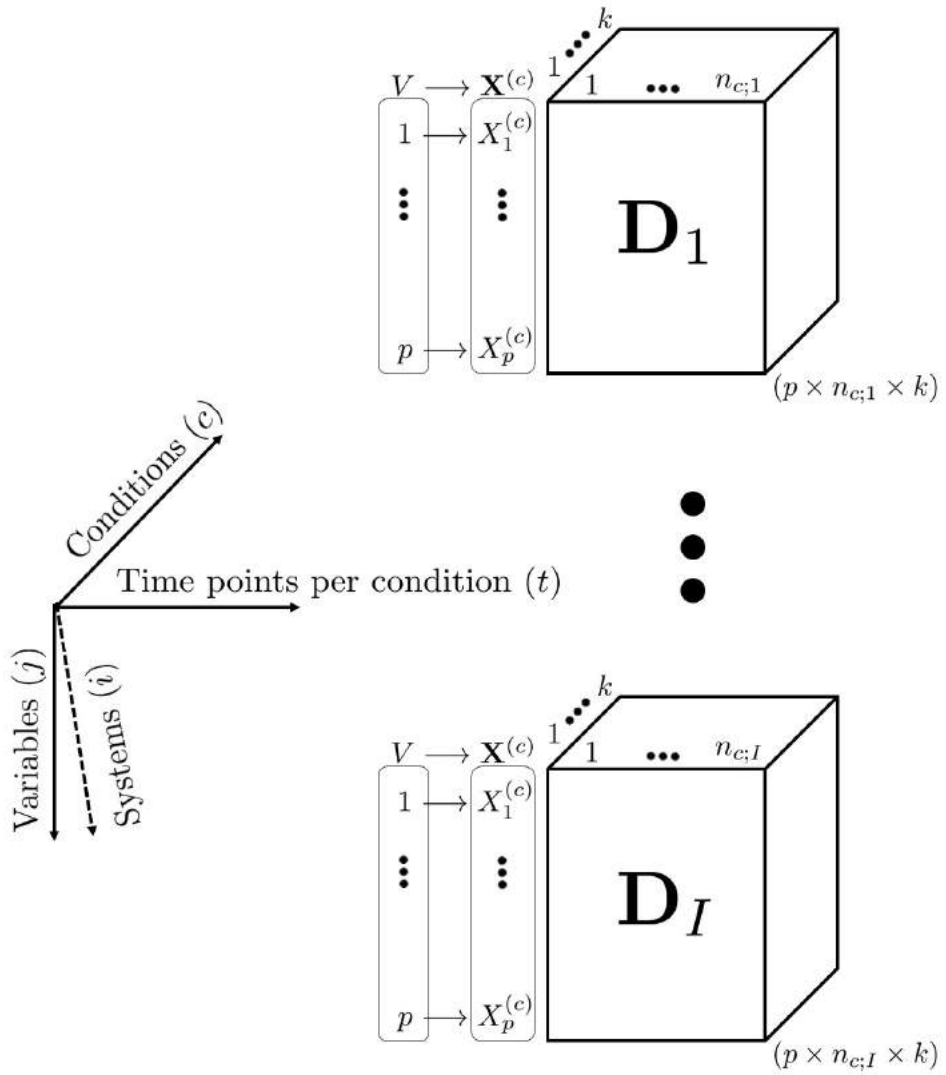


Figure 3.5: Input Data Tensor $\mathbf{D}$. Given I number of related biological systems under observation, let us assume that input data $\mathbf{D}$ is a $(p \times n_{c;i} \times k \times I)$ tensor. Subset of the data corresponding to a particular individual system i is denoted by $\mathbf{D}_i$. The relationship between the given systems would be defined on a case-by-case basis. Each $\mathbf{D}_i$ contains measurements of p system components under k different conditions. For each condition c, there are $n_{c;i}$ independent observations, suppose at $n_{c;i}$ different time points. Total number of observations in $\mathbf{D}_i$ is $\sum_{c=1}^{k} n_{c;i} = N_i$. Similarly, the total number of observations in $\mathbf{D}$ is $\sum_{i=1}^{I} N_i = N$. The system components $v \in V$ are modelled as k random variables $\{X_v^{(c)} : \text{For all } c \in [k]\}$ corresponding to k different conditions; $[n]$ stands for the set of first n natural numbers starting from 1. Then for each of the systems, all its components can be modelled as $(p \times k)$ random variables $\{X_j^{(c)} : j \in [p]\,,\ c \in [k]\}$. Such dataset is typically generated in cohort cell-line studies where k replicates (copies) of diseased tissues are collected from each of the I patients suffering from the same disease. Then each of the patient-specific tissue replicates is treated with a distinct drug. The effect of the drug is monitored for multiple time points.

3.3.4.2 Methods: Reconstructing Summary Network for Each Individual System

In the given data situation, Oates et al. communicate a series of three publications addressing three distinct relationships between the given set of systems. Each publication deals with a more complex relationship than that of its prequel. The relationship between system-specific networks is modelled as a network itself, to be called the **super-network** or **super-structure**. In the super-structure, each node represents a system-specific network. A directed edge between two nodes represents parent-child relationship between the corresponding system-specific networks. All three publications assume that the super-structure is a tree.

- The first publication in the trilogy, Oates et al. (2014) , assumes that all system-specific networks reside at the same level of the tree, whose topology is known a prior.
- Oates and Mukherjee (2014) , the second publication, considers that the system-specific networks reside at different levels of the tree. The tree topology is again known a prior.
- The third publication, Oates et al. (2015) , generalises further by assuming that there is no prior knowledge about the tree topology.

In each of the publications, a distinct variant of Bayesian Network (BayesNet) is used to model a system-specific network. Among them, the most widely-used variant is Dynamic Bayesian Network (DBN) (Murphy, 2002). This is a special class of BayesNet suitable for time-series modelling (Figure 3.6).

Limitation(s)

- Dynamic Bayesian Network (DBN) has some inherent weaknesses. The search space of candidate networks grows exponentially with the number of variables under study (Kim et al., 2014, p. 215). Secondly, it tends to ignore transient edges which are present only for one or few temporal transitions in the unrolled representation. In Song et al. (2009a), this time-homogeneity constraint is stated as the 'Achilles' heel' of DBN. Because in biological signalling networks, such interactions can be of high significance. For example, a short-lived biological regulation can turn on a massive chain of long-lasting regulatory events. Missing the originating transient event would deprive the researchers from a key insight into the concerned system. Especially, in medicine, such insights may result in a highly effective therapy.
- The methods, reviewed in this section, come up with an accurate static summary network, at the best. They do not provide any lead towards learning the temporal progression of the system-specific signalling networks. For every system, such summarisation mechanism across conditions comes at the cost of valuable condition-specific information. Suppose, a cohort study infects a group of volunteers with a particular virus. Some of them get infected, while others stay immune. Currently discussed methods may answer what are the edges that remain active only in the infected sub-group during most of the observed time period. However, they can not identify the edges that become active only during the transition from the healthy stage to the early-disease stage in the infected sub-group but remain

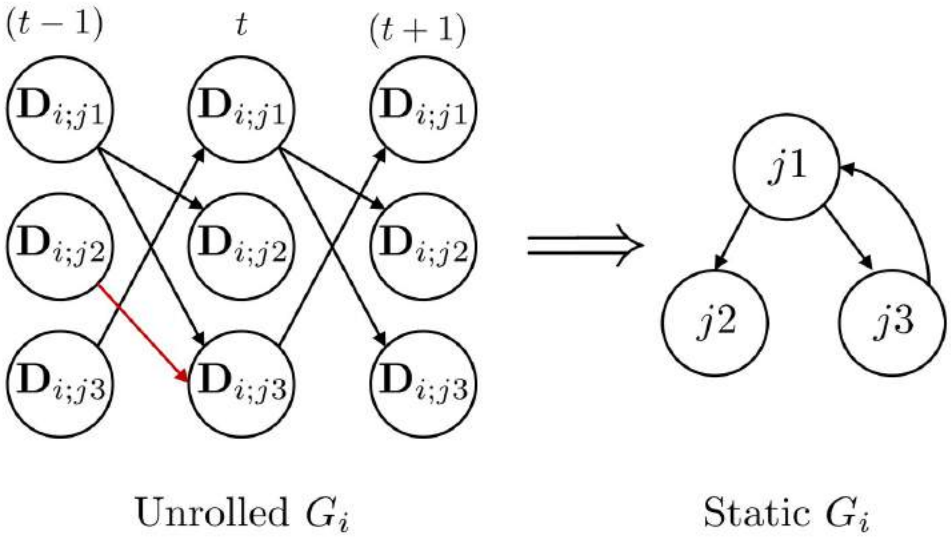


Figure 3.6: Dynamic Bayesian Network (DBN) Modelling. First, an unrolled DBN is inferred for each system $i \in [I]$. The measurements of variable $j1$ in system i at time point $(t-1)$ is represented as node $\mathbf{D}_{i;j1}$ at time point $(t-1)$ (hereafter, $\mathbf{D}_{i;j1}\,(t-1)$). An edge from $\mathbf{D}_{i;j1}\,(t-1)$ to $\mathbf{D}_{i;j2}\,(t)$ implies that the observations of $j2$ at time point t is not conditionally independent of that of $j1$ at the previous time point. After the unrolled DBN reconstruction is completed, it is summarised into a static DBN following some rule, for example - add an edge $(j1, j2)$ in static G_i if there is an edge $(\mathbf{D}_{i;j1}, \mathbf{D}_{i;j2})$ in at least 80% of the temporal transitions (total number of temporal transitions = total number of time points − 1). For example, the edge $(j2, j3)$ is not added to the static DBN because the edge $(\mathbf{D}_{i;j2}, \mathbf{D}_{i;j3})$ (the red arrow) appears only in 50% of the total transitions in the unrolled DBN. It can be noted that static DBN allows cycles unlike BayesNet.

unobserved in the immune sub-group. Knowledge of such disease on-setting regulations can lead to an efficacious cure. In the next section, we review the methods that attempt to identify such regulatory events.

3.3.4.3 Methods: Reconstructing Multiple Time-varying Networks for Each Individual System

Temporal progression modelling is a long-standing challenge in biology and translational medicine (`https://en.wikipedia.org/wiki/Translational_medicine`). It pursues identification of the key system components and their causal roles in development of a disease inside a host system or development of an organism. Given multiple hosts with different disease outcomes, disease progression models help us to distinguish the regulatory events whose presence is a necessary condition for a system to transit from the healthy stage to the disease stage.

With the purpose mentioned above, two major methods, known as **TESLA** (Ahmed and Xing, 2009) and **KELLER** (Song et al., 2009a) are proposed during the same time period. Both of them follow the heuristic that temporally adjacent networks are expected to have more edges in common than the temporally distal networks. TESLA assumes that the temporal changes occur in bursts, like - a discrete function, and produce mul-

tiple discontinuously time-varying sparse Markov Random Fields (MRFs). It follows an ℓ_1-regularised logistic regression formalism to pose a standard convex-optimisation problem that can be solved by any generic solver, for even thousands of variables (Ahmed and Xing, 2009, Equation 2). In addition, it offers user-defined temporal resolution i.e. how many consecutive time points to be grouped together to produce each network. On the other hand, KELLER, assumes that the temporal changes occur smoothly, like - a continuous function, and reconstructs multiple smoothly time-varying sparse undirected networks through kernel weighted ℓ_1-regularised logistic regression. It is directly extended in Xing et al. (2010) which models role of each system component as a time-varying mixed-membership vector that allows the component to interact differently with different components under the same condition as well as interact in a different manner with the same component under distinct conditions. Both TESLA and KELLER are validated with the prior knowledge on a time-series gene expression dataset containing four different developmental stages in Drosophila melanogaster (Dm) (commonly known as fruit fly) life cycle.

The methods, mentioned above, are criticised owing to the use of weaker conditional independence models, like - MRF. Song et al. (2009b) address this issue by proposing a kernel re-weighted ℓ_1-regularised autoregressive structure learning algorithm that reconstructs smoothly time-varying sparse Dynamic Bayesian Networks (DBNs). It is appropriately named as **Time-varying DBNs (TV-DBNs)**. In addition, its kernel re-weighting mechanism aggregates observations from adjacent time points when the amount of data available at a time point is scarce. Due to that, TV-DBN can reconstruct one network for each time point, if required, even in limited data situations. It is used for differential analysis of the datasets collected from multiple human participants in an Electroencephalogram (EEG) motor imagining task. The reconstructed networks are able to provide a causal interpretation that supports the experimental outcome, for example - the networks corresponding to the high-performing participants get denser as the time progresses implying increased interactivity among different brain regions causing the improvement in performance. It leads one step towards understanding how information is processed differently in higher performing individuals.

However, TV-DBN suffers from the inherent time-homogeneity issue of the DBN. Four proposals are reviewed next that address this issue by introducing different variants of inhomogeneous DBNs. Robinson and Hartemink (2009) assume that the changes are discontinuous; each point of discontinuity is called a change-point (in case of continuous changes, local optima or any point can be selected as a change-point), representing transition from one phase to the next. The time-series data is divided into multiple time segments delimited by the change-points. Then a DBN is fitted into each time segment. In addition, a commonality penalty is added to regularise differences between the networks of adjacent time segments. Within the same time segment, network topology (i.e. edge relationships) and parameter values (i.e. edge weights) remain static. On the other hand, Grzegorczyk and Husmeier (2009) reconstruct smoothly time-varying DBNs; each of them is generated from a common parent network, based on the observations at the time points covered by that DBN. There could be multiple networks within a time segment; they must have the same network topology but the parameter values may change in a continuous fashion. Lèbre (2007); Lèbre et al. (2010) add even more flexibility by proposing an algorithm named 'Auto Regressive TIme VArying models' (ARTIVA). This algorithm allows the networks within the same time segment to vary in network topology as well. Grzegorczyk and Husmeier (2011) criticise the first approach to be over-restrictive as change-points are decided based on the whole network. At the same time, they criticise the second and third approaches for their over-flexibility as change-points are specific to each vertex. The criticisms are supported by the fact

that, in dynamic biological systems, different system components may get influenced differently by a process whereas there are some processes, such as - morphogenesis, that influence all system components identically. Being true to their argument, Grzegorczyk and Husmeier propose a fourth approach that strikes a balance between the two extreme paradigms. This method is known as **Time-varying Gene Regulatory Networks (TV-GRNs)**. It uses a Bayesian clustering approach that clusters the vertices in a data-driven manner. Then change-points are decided for each cluster separately. It allows system components belonging to different clusters to be influenced differently by a temporal process while the system components in the same cluster get influenced identically.

In theory, TV-GRN provides adequate flexibility to adjust the temporal resolution; but the modelling accuracy is highly dependent on the clustering scheme which, as Grzegorczyk and Husmeier recommend, is a research opportunity. Moreover, outputted networks only vary in edge parameters. As a consequence, TV-GRN is unable to reconstruct time-varying structures. Therefore, ARTIVA remains the most viable alternative for reconstructing time-varying GRNs till this point.

In line with TV-GRN, Dondelinger et al. argue that the flexibility of ARTIVA may lead to over-fitting, when the number of measurements per gene at each time point is much less than the total number of genes Dondelinger et al. (2013). Hence, they propose 'Information sharing' or 'coupling' between time-interval specific GRN estimators. The proposed framework is categorised into two classes: hard coupling and soft coupling, depending upon the strength of coupling i.e. the expected similarity between time-interval specific GRNs. For hard coupling, two algorithms, TVDBN-bino-hard and TVDBN-exp-hard, are introduced. These algorithms assume that the expression of each gene follows a binomial and an exponential distribution, respectively. Similarly, for soft coupling, two more algorithms, TVDBN-bino-soft and TVDBN-exp-soft, are proposed. An unconstrained (no 'Information sharing') variant, called TVDBN-0, is also proposed. This variant is same as ARTIVA, except in the internal sampling strategies Dondelinger et al. (2013). Dondelinger et al. conclude that 'Information sharing' improves reconstruction when the 'smoothly time-varying assumption' holds; the said assumption states that the true network varies smoothly with time i.e. each GRN structure shares more common edges with its temporally adjacent GRN structures than with the distal ones.

Limitation(s) TVDBN-bino-hard, TVDBN-bino-soft, TVDBN-exp-hard and TVDBN-exp-soft provide a framework to tackle the situation where the number of measurements per gene at each time point is much less than the total number of genes. However, the framework requires the smoothly time-varying assumption. This assumption is not required by ARTIVA. Nevertheless, ARTIVA is a computationally expensive algorithm. It requires approximately 5 minutes per gene to reconstruct GRNs from a dataset with 20 time points and 5 measurements per gene at each time point on a 2.66 GHz CPU having a 4 GB RAM Lèbre et al. (2010). With that speed, the time frame necessary for ARTIVA to scale up to large-scale datasets may be considered prohibitive ($\simeq$ 87 days for $25,000$ genes). Hence, developing algorithms that do not require the smoothly time-varying assumption as well as computationally efficient can be a worthwhile challenge.

3.4 Chapter Summary: Abridged Literature Survey

In the previous chapter (Chapter 1), we discuss the motivation behind constructing network-based computational models for studying the progression of dynamic biological systems. With rapid advancements in experimental technologies, datasets are increas-

ingly getting larger and richer. In this chapter, it is analysed how the complexity of the progression models increases with that of the datasets. The major proposals that address these challenges are reviewed. Since progression modelling of dynamic biological systems is a data-driven scientific endeavour, it is meaningful to categorise the models and corresponding reconstruction algorithms based on the following questions:

- **Model Selection:** Given a dataset with certain properties and prior knowledge, if any, is there any model that can capture the underlying dependency structure which has generated the data?
- **Selection of Reconstruction Algorithm:** Given the properties of the data and the chosen model, which reconstruction algorithm would be most suitable?

A brief graphical summary of the proposals, categorised based on the aforementioned criteria, is given in Figure 3.7 .

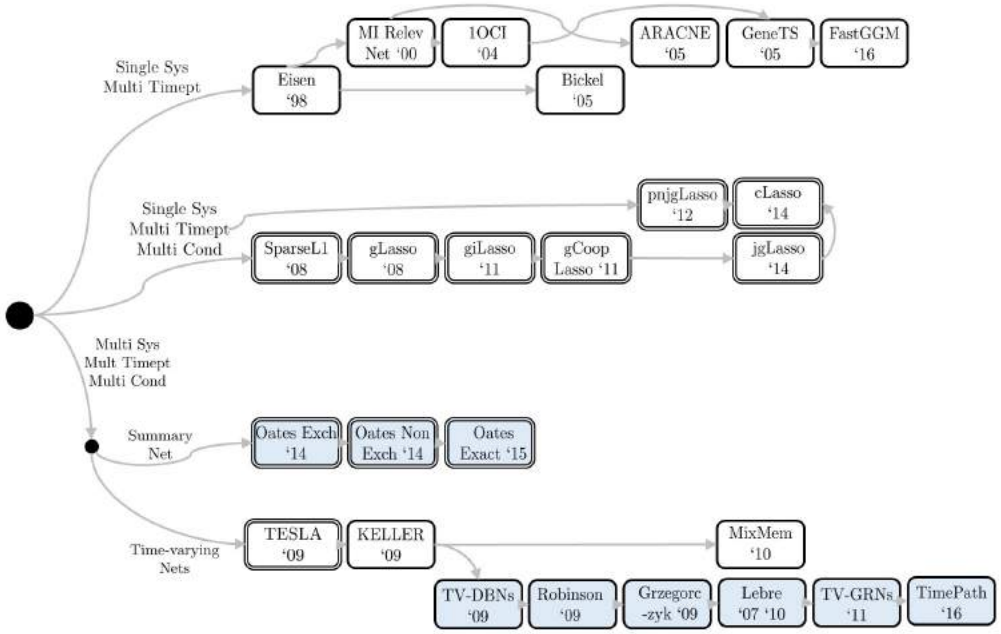


Figure 3.7: A graphical summary of the literature review. A single-bordered white box indicates that the corresponding method generates undirected network using an Independent Network Inference (INI) algorithm. On the other hand, the double-border signifies that a Joint Network Inference (JNI) algorithm is employed. A blue box implies that the generated network or networks is a directed graph. A directed edge between two boxes signifies that the end box is an extension of the start box. Reference to each publication is given in Table 3.1.

Table 3.1: References for Figure 3.7

Ref ID in Fig 3.7	**Citation Key**	**Remark**
Eisen '98	Eisen et al. (1998)	Hypothesises that co-expression implies involvement in the same function
MI Relev Net '00	Butte and Kohane (2000)	Mutual Information Relevance Networks
1OCI '04	Magwene and Kim (2004); De La Fuente et al. (2004); Wille and Bühlmann (2006); Wille et al. (2004)	First Order Conditional Independence Models
Bickel '05	Bickel (2005)	Introduces 'Time-lag' between regulator and regulatee
ARACNE '05	Basso et al. (2005); Margolin et al. (2006)	Reduces false positive edges with the help of Data Processing Inequality principle
GeneTS '05	Schäfer and Strimmer (2005)	Solves Gaussian Graphical Model inference when number of variables $\gg$ sample size
FastGGM '16	Wang et al. (2016)	R package for inferring Gaussian Graphical Model with $10,000$ nodes in reasonable time
SparseL1 '8	Banerjee et al. (2008)	Inferring sparse networks with ℓ_1 regularisation
gLasso '08	Friedman et al. (2008)	Graphical Lasso: First use of Lasso framework for inferring graphical models
giLasso '11	Chiquet et al. (2011)	Graphical Intertwined Lasso: Graphical Lasso with an intertwined estimation of the covariance matrix
gCoopLasso '11	Chiquet et al. (2011)	Graphical Cooperative Lasso: Imposes the constraint that a system component up-regulating another component in one condition, is less likely to down-regulate the latter in another condition
pnjgLasso '12	Mohan et al. (2012)	Perturbed Node Joint Graphical Lasso: Introduces commonality constraint between condition-specific covariance matrices
jgLasso '14	Danaher et al. (2014)	Joint Graphical Lasso: Tightens the commonality constraint in Graphical Lasso by replacing ℓ_2 regularisation with ℓ_1 regularisation
cLasso '14	Villa-Vialaneix et al. (2014)	Consensus Lasso: Composes a single objective function that satisfies sparsity as well as commonality constraints

continued ...

. . . continued

Ref ID in Fig 3.7	Citation Key	Remark
Oates Exch ‘14	Oates et al. (2014)	Reconstructs individual-specific Dynamic Bayesian Networks for exchangeable individuals
Oates Non Exch ‘14	Oates and Mukherjee (2014)	Reconstructs individual-specific Dynamic Bayesian Networks for non-exchangeable individuals
Oates Exact ‘15	Oates et al. (2015)	Performs exact estimation of individual-specific Dynamic Bayesian Networks when there is no prior knowledge about the relationship between the given individuals
TESLA ‘09	Ahmed and Xing (2009)	Reconstructs multiple discontinuously time-varying Markov Random Fields for each individual
KELLER ‘09	Song et al. (2009a)	Reconstructs multiple continuously (smoothly) time-varying Markov Random Fields for each individual
TV-DBNs ‘09	Song et al. (2009b)	Reconstructs multiple smoothly time-varying Dynamic Bayesian Networks for each individual
Robinson ‘09	Robinson and Hartemink (2009)	Reconstructs multiple discontinuously time-varying inhomogeneous Dynamic Bayesian Networks for each individual. Each network represents a time segment in the given time-series data for that individual. Within the same time-segment, network topology and edge parameters remain static.
Grzegorczyk ‘09	Grzegorczyk and Husmeier (2009)	Reconstructs multiple smoothly time-varying inhomogeneous Dynamic Bayesian Networks for each individual. Each time segment may have multiple networks with the same network topology but smoothly varying edge parameters.
Lebre ‘07 ‘10	Lèbre (2007); Lèbre et al. (2010)	Reconstructs multiple smoothly time-varying inhomogeneous Dynamic Bayesian Networks for each individual. Each time segment may have multiple networks with varying network topology and smoothly varying edge parameters.
MixMem ‘10	Xing et al. (2010)	Models role of each system component as a time-varying mixed membership vector

continued . . .

. . . continued

Ref ID in Fig 3.7	Citation Key	Remark
TV-GRNs '11	Grzegorczyk and Husmeier (2011)	Reconstructs multiple time-varying Gene Regulatory Networks for each individual. Clusters the system components. Restricts time-variation of the system components within the same cluster to follow the same pattern.
TimePath '16	Jain et al. (2016)	Reconstructs multiple time-varying partially directed cellular signalling networks. It integrates time-course gene expression data with static protein-protein interaction data and protein-DNA interaction data to find out key proteins/genes and major pathways involved in HIV-1 immune response.

Chapter 4

Problem Formulation

In the previous chapter, we critically review the existing reconstruction algorithms (Chapter 3). In this chapter, we select a subset of the challenges faced by these algorithms as the objective of this thesis.

The algorithms of our interest reconstruct time-varying Gene Regulatory Networks (GRNs) from time-series gene expression datasets. Among these algorithms, the ones that offer state-of-the-art frameworks are: *ARTIVA*, *TVDBN-0*, *TVDBN-bino-hard*, *TVDBN-bino-soft*, *TVDBN-exp-hard* and *TVDBN-exp-soft*. In the following sections, we conduct a comparative study of the aforementioned algorithms to identify their limitations. Consequently, some of the limitations are proposed to be overcome in this thesis.

4.1 Notations

4.1.1 Input: Time-series Gene Expression Dataset

Suppose that the given dataset $\mathcal{D}$ is comprised of a set of time series $\mathcal{S} = \{s_1, \ldots, s_S\}$ of gene expression data (Figure 4.1). Each time series contains the expression levels of a set of genes $\mathcal{V} = \{v_1, \ldots, v_V\}$ at T consecutive time points $\mathcal{T} = \{t_1, \ldots, t_T\}$. It is assumed that there are no missing values in any time series. In other words, each time series is a complete time series of T time points. Notation $\mathcal{D}_{(\mathcal{X};\mathcal{Y};\mathcal{Z})}$ is used to denote the observed values of genes $\mathcal{X}$ at time points $\mathcal{Y}$ in time series $\mathcal{Z}$. Hence, $\mathcal{D}_{(\mathcal{X};\mathcal{Y};\mathcal{Z})} \subseteq \mathcal{D}$ where $\mathcal{X} \subseteq \mathcal{V}, \mathcal{Y} \subseteq \mathcal{T}, \mathcal{Z} \subseteq \mathcal{S}$.

4.1.2 Output: Time-varying Gene Regulatory Networks

Given dataset $\mathcal{D}$, the objective is to reconstruct a temporally-ordered sequence of GRNs $\mathcal{G} = \left(G^{(1)}, \ldots, G^{(T-1)}\right)$ (Figure 4.2). Here, each $G^{(p)} (\in \mathcal{G})$ is a time interval specific GRN. Thus, $G^{(p)}$ represents the gene regulatory events occurred during the time interval between time points t_p and $t_{(p+1)}$. It is a directed unweighted network on the $(2 \times V)$ nodes $\{v_i\_t_q \colon v_i \in \mathcal{V}, t_q \in \{t_p, t_{(p+1)}\}\}$. Each node $v_i\_t_q$ represents a distinct random variable. Expression of gene v_i at time point t_q is modelled as random variable $v_i\_t_q$. Therefore, the observed expression values of v_i at time point t_q in S separate time series are considered as S observed values of $v_i\_t_q$.

In this thesis, the underlying gene regulation process is assumed to be first-order Markovian (Friedman et al., 1998, p. 140, Section 2). The assumption states that the expression of a gene at a particular time point only depends upon the expressions of

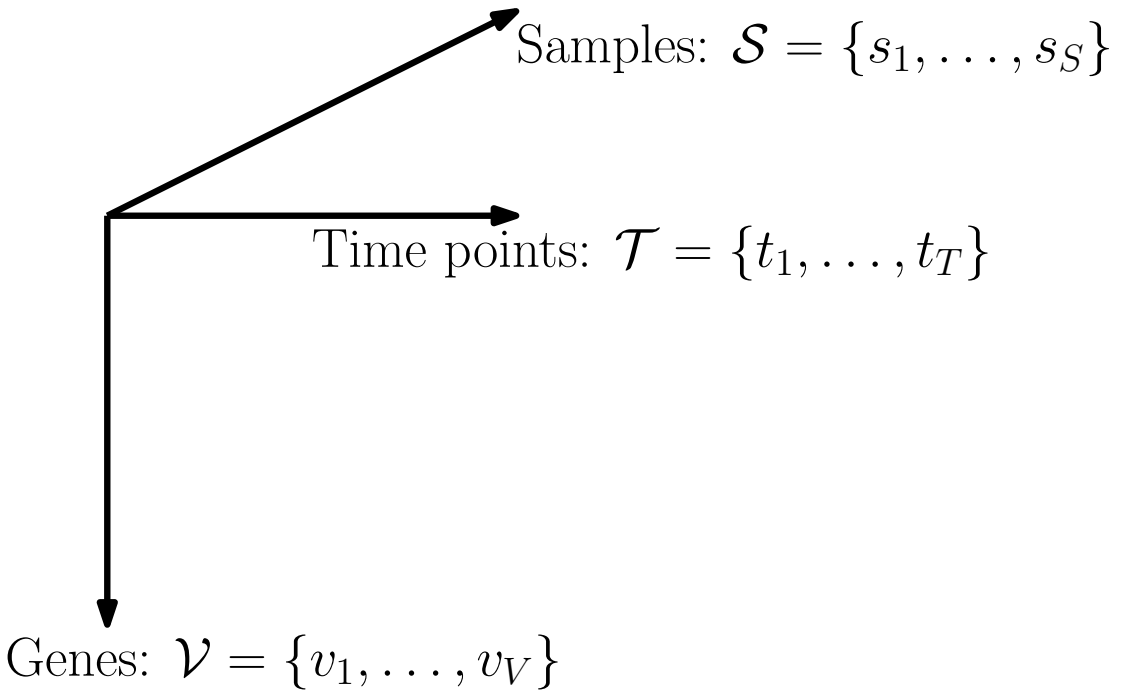


Figure 4.1: Input time-series gene expression data $\mathcal{D}$ is a three dimensional tensor with the dimensions (V genes, T time points, S time series). $\mathcal{D}$ is comprised of a set of time series $\mathcal{S} = \{s_1, \ldots, s_S\}$ of gene expression data. Each time series contains the expression levels of a set of genes $\mathcal{V} = \{v_1, \ldots, v_V\}$ at T consecutive time points $\mathcal{T} = \{t_1, \ldots, t_T\}$. It is assumed that there are no missing values in any time series. In other words, each time series is a complete time series of T time points. Notation $\mathcal{D}_{(\mathcal{X};\mathcal{Y};\mathcal{Z})}$ is used to denote the observed values of genes $\mathcal{X}$ at time points $\mathcal{Y}$ in time series $\mathcal{Z}$. Hence, $\mathcal{D}_{(\mathcal{X};\mathcal{Y};\mathcal{Z})} \subseteq \mathcal{D}$ where $\mathcal{X} \subseteq \mathcal{V}, \mathcal{Y} \subseteq \mathcal{T}, \mathcal{Z} \subseteq \mathcal{S}$.

its regulators at the previous time point. This is a realistic assumption. The time interval between two consecutive time points in a dataset is usually sufficiently large for regulators to have an effect on the target gene's expression. For example, the time interval is around one hour in a widely-used human dataset (Zaas et al., 2009). This interval is adequately large for regulators to effect their target gene's expression in human cells. For instance, in human Fibroblast cells, regulators require only half an hour to have an effect on their target gene's expression (Alon, 2006, Table 2.1).

However, the time required for a regulator to effect its target is not always known. As a result, the ideal time interval for data collection can not be determined. In that case, time intervals in the data may not honour the actual effecting time. For such data, higher order dependencies can be considered. If considered, finding out those dependencies increases computational complexities. To avoid higher complexities, the first-order Markovian assumption is widely used in the relevant literature (Lèbre et al., 2010; Dondelinger et al., 2013).

From the first-order Markovian assumption, it follows that the expression of v_j at time point $t_{(p+1)}$ depends only upon its regulators at time point t_p. If the expression of v_i at time point t_p has a regulatory effect on that of v_j at time point $t_{(p+1)}$, it is represented as a directed edge $\left(v_i\_t_p, v_j\_t_{(p+1)}\right)$ in the output time-varying GRNs (Figure 4.2).

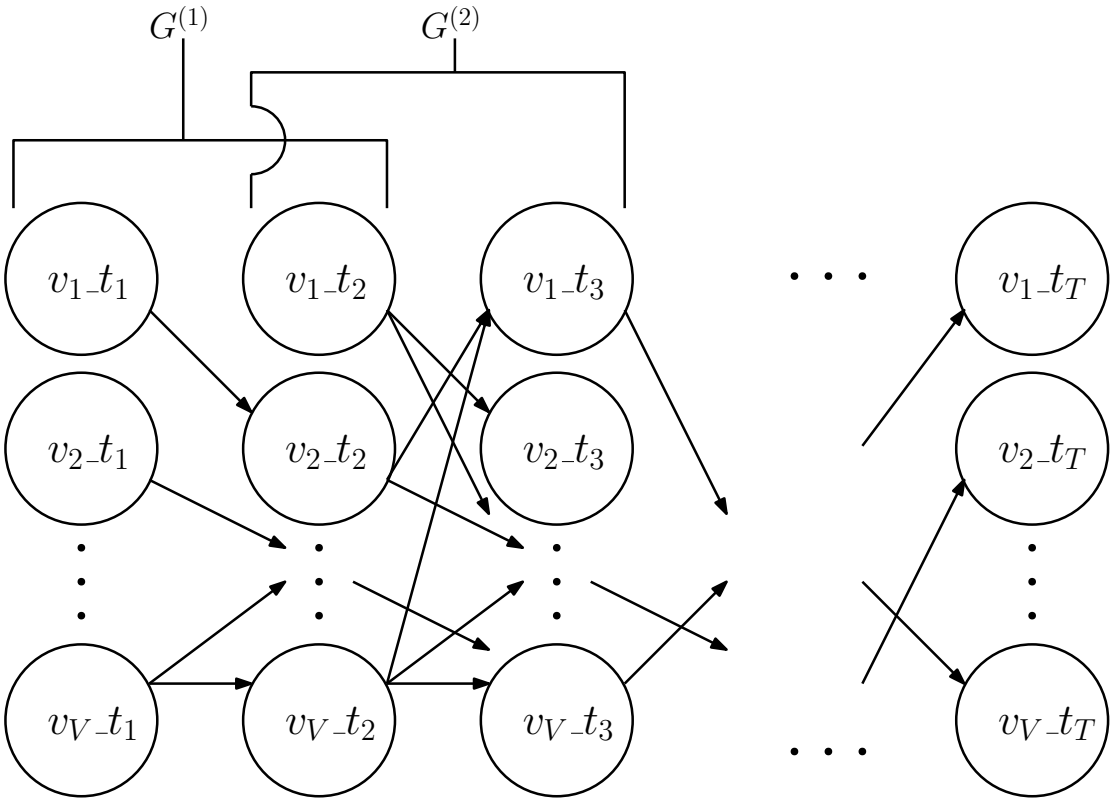


Figure 4.2: Output time-varying GRNs $\left(G^{(1)}, \ldots, G^{(T-1)}\right) = \mathcal{G}$ is a sequence of directed unweighted networks. Here, $G^{(p)} (\in \mathcal{G})$ represents the gene regulatory events occurred during the time interval between time points t_p and $t_{(p+1)}$. It consists of $(2 \times V)$ nodes $\{v_i\_t_q \colon v_i \in \mathcal{V}, t_q \in \{t_p, t_{(p+1)}\}\}$. There exists a directed unweighted edge $\left(v_i\_t_p, v_j\_t_{(p+1)}\right)$ if and only if v_i regulates v_j during time interval $\left(t_p, t_{(p+1)}\right)$.

4.2 Benchmark Datasets

A real gene expression dataset with known true underlying GRNs is the coveted choice of dataset for evaluating GRN modelling algorithms. To that end, Marbach et al. (2009) design three sets of realistic GRN structures for different model organisms with 10, 50 and 100 genes. Then for each of these in silico GRNs, they choose an appropriate dynamical model and generate a dataset through simulation (Marbach et al., 2010, p. 6290, Section 'Simulation of Expression Data.'; Supplementary Information, Section 'SI Methods. Gene network model.'). These datasets are made publicly available as benchmarks for assessing and comparing the modelling algorithms through DREAM3 In Silico Network Challenge DREAM3; Prill et al. (2010). In each dataset, gene expressions are normalized so that the maximum gene expression value in a data file is one. Among these datasets, the 'Yeast1' time-series datasets are chosen for the purpose of this thesis. It comprises of two sets of datasets: one noiseless and the other noisy (resulted from adding Gaussian noise to the noiseless datasets). Each set contains three datasets as summarized in Table 4.1 . It can be noted that the true networks are summary GRNs.

Table 4.1: A Summary of the chosen DREAM3 Datasets. Here, V = number of genes, T = number of time points, and S = number of time series.

Dataset (noiseless)	**V**	**T**	**S**	**No. of True Edges**
Ds10	10	21	4	10
Ds50	50	21	23	77
Ds100	100	21	46	166
Dataset (noisy)	**V**	**T**	**S**	**No. of True Edges**
Ds10n	10	21	4	10
Ds50n	50	21	23	77
Ds100n	100	21	46	166

4.3 Evaluation Metrics

The true networks are summary GRNs; on the other hand, the outputs of reconstruction algorithms are time-varying GRNs. Hence, the output set of networks $\mathcal{G}$ for each algorithm is converted ('rolled up') into an equivalent single network G by the following algorithm: Add a directed edge from v_i to v_j in G if there exists at least one edge from $v_i\_t_p$ to $v_j\_t_{(p+1)}$ in $\mathcal{G}$ for any $t_p, t_{(p+1)} \in \mathcal{T}$. For DREAM3 synthetic datasets, self-loops (if any) are removed from the rolled network G since the true networks do not contain self-loops.

Given a dataset and the corresponding true network, the following metrics are used to evaluate the learning power of an algorithm. True Positive (TP) and False Positive (FP) stand for the number of edges correctly predicted and number of edges incorrectly predicted, respectively. On the other hand, True Negative (TN) and False Negative (FN) represent the number of non-edges (absence of edges in the true network) correctly predicted and number of non-edges incorrectly predicted, respectively.

- True Positive Rate (TPR) (or Recall):
  $TPR = TP/(TP + FN)$.
  It measures how good an algorithm is in capturing the correct edges.
- Positive Predictive Value (PPV) (or Precision):
  $PPV = TP/(TP + FP)$.
  It measures how good an algorithm is in rejecting the incorrect edges.
- F1-Score:
  $F1 = (2 \times PPV \times TPR) \,/\, (PPV + TPR)$. When PPV and TPR are zeros, $F1$ is considered zero. F1-score is simply the harmonic mean of recall and precision. As a result, it is a measure of an algorithm's ability to balance recall and precision. We utilise F1-score to evaluate the overall correctness of the time-varying GRNs reconstructed by an algorithm.

4.4 Comparative Study of the Existing Algorithms

In this section, we conduct a comparative study of the existing reconstruction algorithms. For the study, we apply them on benchmark datasets Ds10n, Ds50n and Ds100n. Consequently, the correctnesses of their reconstructed networks are evaluated.

4.4.1 Implementations

The source code of *ARTIVA* is publicly available as an R package with the same name (ARTIVA, version: 1.2.3). The source codes of *TVDBN-0*, *TVDBN-bino-hard*, *TVDBN-bino-soft*, *TVDBN-exp-hard* and *TVDBN-exp-soft* are publicly available as an R package named 'EDISON' ((EDISON), version: 1.1.1).

Experiments are performed on an Intel® computing server with the following configuration:

- Architecture: x86_64
- CPUs: Two Intel® Xeon® X5675 @ 3.07GHz CPUs
- Main Memory: 31 GB
- Swap Space: 34 GB
- Cache: {L1d cache: 32 KB, L1i cache: 32 KB, L2 cache: 256 KB, L3 cache: 12288 KB}
- Secondary Storage: 4.1 TB
- Operating System: Ubuntu 12.04.5 LTS (Codename: Precise)

4.4.2 Results

We observe that *ARTIVA* substantially outperforms other algorithms in F1-score for two of the three benchmark datasets (Table 4.2). However, such superiority in correctness comes at the cost of computational time (Table 4.3). *ARTIVA* consumes around 1.5 days to process dataset Ds100n which contains only a hundred genes.

It can be noted that the implementations of *TVDBN-exp-hard* and *TVDBN-exp-soft* raise errors in all the experiments. In a personal communication with us, the maintainer of the corresponding package hints at a potential bug. Hence, the results of *TVDBN-exp-hard* and *TVDBN-exp-soft* are not reported here.

Table 4.2: The F1-scores of the Existing Reconstruction Algorithms for Benchmark Datasets Ds10n, Ds50n and Ds100n. The numerical values are rounded off to three decimal places. For each dataset i.e. column, the best value is boldfaced.

Algorithm	**Ds10n**	**Ds50n**	**Ds100n**
ARTIVA	0	**0.082**	**0.083**
TVDBN-0	0	0.049	0.021
TVDBN-bino-hard	0.111	0.044	0.035
TVDBN-bino-soft	**0.190**	0.058	0.024

4.5 Problem Statement

ARTIVA's runtime is a major concern for large-scale datasets with hundreds to thousands of genes (Zaas et al., 2009). At the same time, rapid advancements of data-acquisition technologies are making it possible to produce increasingly larger datasets.

Table 4.3: The runtime of the Existing Reconstruction Algorithms for Benchmark Datasets Ds10n, Ds50n and Ds100n. The numerical values are rounded off to three decimal places.

Algorithm	**Ds10n**	**Ds50n**	**Ds100n**
ARTIVA	10 m 20 s	4 h 30 m 15 s	31 h 52 m 54 s
TVDBN-0	2 m 24 s	11 m 59 s	52 m 17 s
TVDBN-bino-hard	2 m 15.2 s	9 m 38 s	2 h 53 m 32 s
TVDBN-bino-soft	2 m 14.6 s	8 m 8 s	17 m 20 s

Initiatives like The Precision Medicine Initiave (PMI) (`https://www.whitehouse.gov/precision-medicine`) and Google Baseline Study (GBS) (`https://en.wikipedia.org/wiki/Baseline_Study`) are expected to generate such massive datasets. As a consequence, the gap between the desired computational speed and that of *ARTIVA* is becoming wider. In this thesis, we propose to bridge this gap. Therefore, the problem statement can be formalised as follows: The objective of this thesis is to propose algorithms that can

- deliver correctness competitive to that of *ARTIVA*, and
- computational efficiency compatible with large-scale datasets with hundreds to thousands of genes.

4.6 Chapter Summary

In Chapter 2 , we investigate the motivation behind network-based progression modelling of dynamic biological systems. A critical review of the relevant methods is presented in Chapter 3 . That provided us insights required to design better methodology and a range of research challenges. In this chapter, we focus on a subset of the challenges based on feasibility and analytical demand of the contemporary experimental studies. Specifically, the focal point is set on the large-scale studies that monitor hundreds to thousands of genes across time. Availability of these massive datasets makes it an appropriate time to study developmental and disease progression modelling. However, it is found that the existing methods known to provide state-of-the-art correctness can not scale up to such high-dimensional data settings. Therefore, it is necessary to bridge the gap between correctness and efficiency for discovering the knowledge hidden in rich large-scale datasets. The objective of this thesis is to mend that gap.

Chapter 5

Improving Time-efficiency

In the previous chapter, we formulate the objective of this thesis (Chapter 4). The objective is to develop reconstruction algorithms that can:

- deliver correctness competitive to that of *ARTIVA*, which is the state-of-the-art algorithm in terms of correctness; and
- offer computational efficiency compatible with large-scale datasets with hundreds to thousands of genes.

Towards that objective, we propose a time-efficient algorithm in this chapter. The algorithm is called 'an algorithm for reconstructing Time-varying Gene regulatory networks with Shortlisted candidate regulators', in short, *TGS*.

This chapter is organised into multiple sections. In Section 5.1 , we design a novel algorithm. Subsequently, the experimental results are discussed in Section 5.2 . An excerpt along with a pointer to the future work are provided in Section 5.3 . The research contributions are acknowledged in Section 5.4 . Finally, in Section 5.5 , a summary of the chapter is presented.

5.1 Methods

In this section, two algorithms are developed. First, a baseline algorithm is developed for reconstructing time-varying Gene Regulatory Networks (GRNs) from time-series gene expression data. Like ARTIVA (Lèbre et al., 2010, Section 'Conclusions'), the baseline algorithm attempts to reconstruct the time-varying GRNs independently of each other. Therefore, it is compatible with any dataset regardless of whether the smoothly time-varying assumption holds for it or not. Nevertheless, it is time-intensive and hence not suitable for large-scale datasets. Second, a set of heuristic based approximation steps is added to the baseline algorithm to develop the final algorithm. The latter maintains the independently time-varying framework without compromising the time-efficiency.

5.1.1 Development of the Baseline Algorithm

In this section, a conditional independence based baseline algorithm, referred to as Time-varying Bayesian Networks (*TBN*), is designed. Algorithm 8 describes the steps in *TBN*. It takes a discretised complete time-series gene expression dataset $\mathcal{D}$ as input. It is assumed that there are multiple time series ($S > 1$) in $\mathcal{D}$. Then *TBN* reconstructs one GRN $G^{(p)}$ for every time interval $\left(t_p, t_{(p+1)}\right)$, where $1 \leq p \leq (T-1)$ (Figure 4.2).

Each $G^{(p)}$ is modelled as a Bayesian Network (BN) (Markowetz and Spang, 2007, Section 'Bayesian networks'). Absence of a directed edge $\left(v_i\_t_p, v_j\_t_{(p+1)}\right)$ in $G^{(p)}$ implies that the expression level of v_j at time point $t_{(p+1)}$ is conditionally independent of that of v_i at time point t_p, given the expression levels of the genes $\mathcal{V} \setminus \{v_i\}$ at time point t_p. Biologically, it signifies that the expression level of v_i at time point t_p has no regulatory effect on that of v_j during time interval $(t_p, t_p + 1)$. On the other hand, presence of that edge signifies that there is a non-zero probability that v_i's expression level at t_p has affected that of v_j during the $(t_p, t_p + 1)$ time interval.

TBN employs a BN structure learning algorithm (Murphy, Section 'Structure learning') to learn every $G^{(p)}$ from $\mathcal{D}_{\left(\mathcal{V};\{t_p,t_{(p+1)}\};\mathcal{S}\right)}$. Therefore, the problem of learning $(T-1)$ time-varying GRNs in $\mathcal{G}$ gets decomposed into $(T-1)$ independent BN structure learning problems. For learning an exact BN structure, *Bene* is the state-of-the-art algorithm w.r.t. time complexity and scalability, to the best of our knowledge (Silander and Myllymäki, 2006). Hence, *TBN* with *Bene* is chosen as the baseline for developing a novel algorithm.

In *TBN* (Algorithm 8) , BIC scoring function (Markowetz and Spang, 2007, Section 'Bayesian information criterion (BIC)') is used with *Bene* to compute scores of the candidate regulator sets. There exist some other scoring functions that can be used with *Bene*, e.g. BDe. Among all available scoring functions, BIC and BDe are compared w.r.t. their effects on learning power of *Bene* by Silander et al. (Silander and Myllymäki, 2006, Section 4.4). It is observed that BIC outperforms BDe when number of observations being considered (the value of expression $(S+1)$, to be specific) is below 20. Moreover, the performance of BDe is very sensitive to the chosen value of its hyper-parameter Silander (2017). BIC, on the other hand, does not depend on any hyper-parameters. For these reasons, BIC is considered to be the most suitable scoring function for the current study.

5.1.2 Development of a Novel Algorithm: The TGS Algorithm (short form for 'an algorithm for reconstructing Time-varying Gene regulatory networks with Shortlisted candidate regulators')

From Algorithm 8 , it is found that *TBN*'s time complexity is $T_{\text{TBN}}(V) = (T-1) \times V \times o\left(V^2 2^{(V-2)}\right) = o\left((T-1)\, V^3 2^{(V-2)}\right)$. The time complexity grows exponentially with the number of candidate regulators for each gene, which is V in this case. Therefore, this approach can be made more computationally efficient if a way can be discovered that: (a) generates a significantly shorter list of candidate regulators for each gene, and (b) the amount of time it spends for short-listing candidate regulators is overshadowed by the time gain it brings.

Statistical pairwise association measures fulfil the first criterion. Given sufficient observations on a pair of random variables, they can identify whether there is a statistically significant probability (w.r.t. a predefined significance threshold) that these variables are not associated with each other. Thus the candidate regulators, whose expressions are not statistically associated with that of the regulatee gene, could be identified. Then these regulators can be removed from the candidate regulator set.

A set of fourteen such measures are comparatively studied by Liu et al. Liu (2017) who conclude that Mutual Information (MI) demonstrates superior stability over other measures. MI's potential regulator-regulatee association predictions consistently outperform (Liu, 2017, Section 'Results of Comparison Study' and Figure 3) those of most others across different sizes (different values of V) of benchmark gene expression datasets w.r.t. mean AUC (Area Under Receiver Operating Characteristic Curve). Algorithms

NARROMI and *LBN* utilize MI for short-listing candidate regulators. For each regulatee gene, they calculate its MI with every candidate regulator; then eliminate the candidates with MI lower than a user-defined threshold.

However, the Achilles' heel of this strategy is that the prediction is heavily dependent on the user-defined threshold value (Liu et al., 2016, Section 'Effects of the threshold parameters'). *LBN* determines the threshold for synthetic datasets by performing the predictions multiple times with different threshold values and choosing the one that gives the best prediction. This threshold selection strategy requires the true regulatory relationships to be known a priori so that the quality of a prediction can be measured. A more practical strategy is appointed by Context Likelihood of Relatedness (*CLR*) algorithm Faith et al. (2007) (Algorithm 9) . It reconstructs a weighted MI network [1] over all genes from a gene expression dataset without requiring a user-defined threshold. *CLR* is found to outperform other major MI network reconstruction algorithms (Faith et al., 2007, Figure 2). Furthermore, it requires only $\mathcal{O}\left(V^2\right)$ time for a dataset with V genes.

For the aforementioned reasons, *CLR* is chosen to be a pre-selection step for candidate regulators before more comprehensive selection could be performed by *TBN*. It gives birth to a novel algorithm, which is named *TGS* (short form for 'an algorithm for reconstructing Time-varying Gene regulatory networks with Shortlisted candidate regulators'). A graphical flowchart is presented in Figure 5.1.

TGS (Algorithm 10) has the time complexity $T_{\mathrm{TGS}}(V) = \left(\mathcal{O}\left(V^2\right) + (T-1) \times V \times o\left(M^2 2^{(M-2)}\right)\right) = \left(\mathcal{O}\left(V^2\right) + o\left((T-1)\, V M^2 2^{(M-2)}\right)\right)$. Here, M is the maximum number of neighbours a gene has in the *CLR* network. In theory, $M \leq V$, time complexity of *TGS* is upper bounded by that of *TBN*. Empirically, it is found (Bhardwaj et al., 2010, Figure 2) that each gene is regulated by a small number of regulators with the exception in case of E.coli. For this reason, major BN based algorithms (e.g., the *DBN* implementation in BayesNet Toolbox for MATLAB Murphy (2001)) have variants that allow the user to specify the maximum number of regulators a gene can have. This user-defined value is known as the max fan-in (M_f). For each gene, it reduces the number of candidate regulator sets from $\sum_{m=0}^{V} \binom{V}{m}$ to $\sum_{m=0}^{M_f} \binom{V}{m}$, where $M_f \ll V$. It is further reduced to $\sum_{m=0}^{M_f} \binom{M_f}{m}$ in the variant of *TGS* with the max fan-in restriction (Algorithm 11) . Therefore, for a high-throughput human-genome scale time series gene expression dataset where $(T-1) = o(V)$ and $M_f = o(\lg V)$, the time complexity of *TGS* asymptotically tends towards polynomial while that of *TBN* remains exponential.

$$
\begin{aligned}
T_{\mathrm{TBN}}(V) &= o\left((T-1)\, V^3\ 2^{(V-2)}\right) \\
&= o\left(o(V)\, V^3\ 2^{(V-2)}\right) && \because (T-1) = o(V) \\
&= o\left(V^4\ 2^{(V-2)}\right)
\end{aligned}
$$

[1] An MI network is an undirected graph where two nodes are connected if and only if their pairwise MI is statistically significant. The significance threshold is either user-defined or programmatically computed by the network reconstruction algorithm itself.

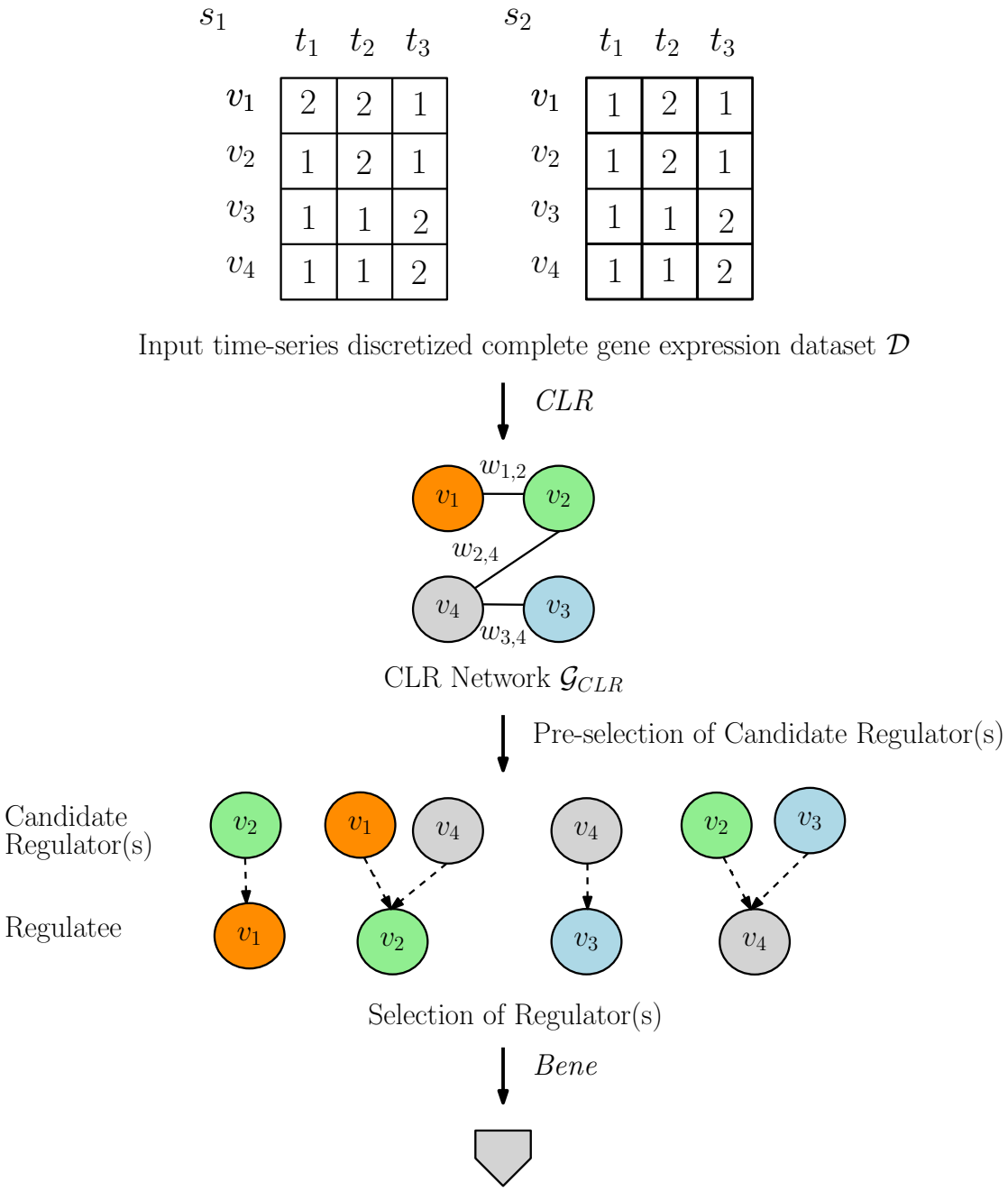


Figure 5.1: Graphical Flowchart (Part 1) of the *TGS* Algorithm. The flowchart is continued in Figure 5.2. For illustration, a dataset $\mathcal{D}$ is considered with four genes $\{v_1, v_2, v_3, v_4\} = \mathcal{V}$ and two time series $\{S_1, S_2\} = \mathcal{S}$. Each time series has three time points $\{t_1, t_2, t_3\} = \mathcal{T}$. $\mathcal{D}$ is discretised into two discrete levels, represented by $\{1, 2\}$.

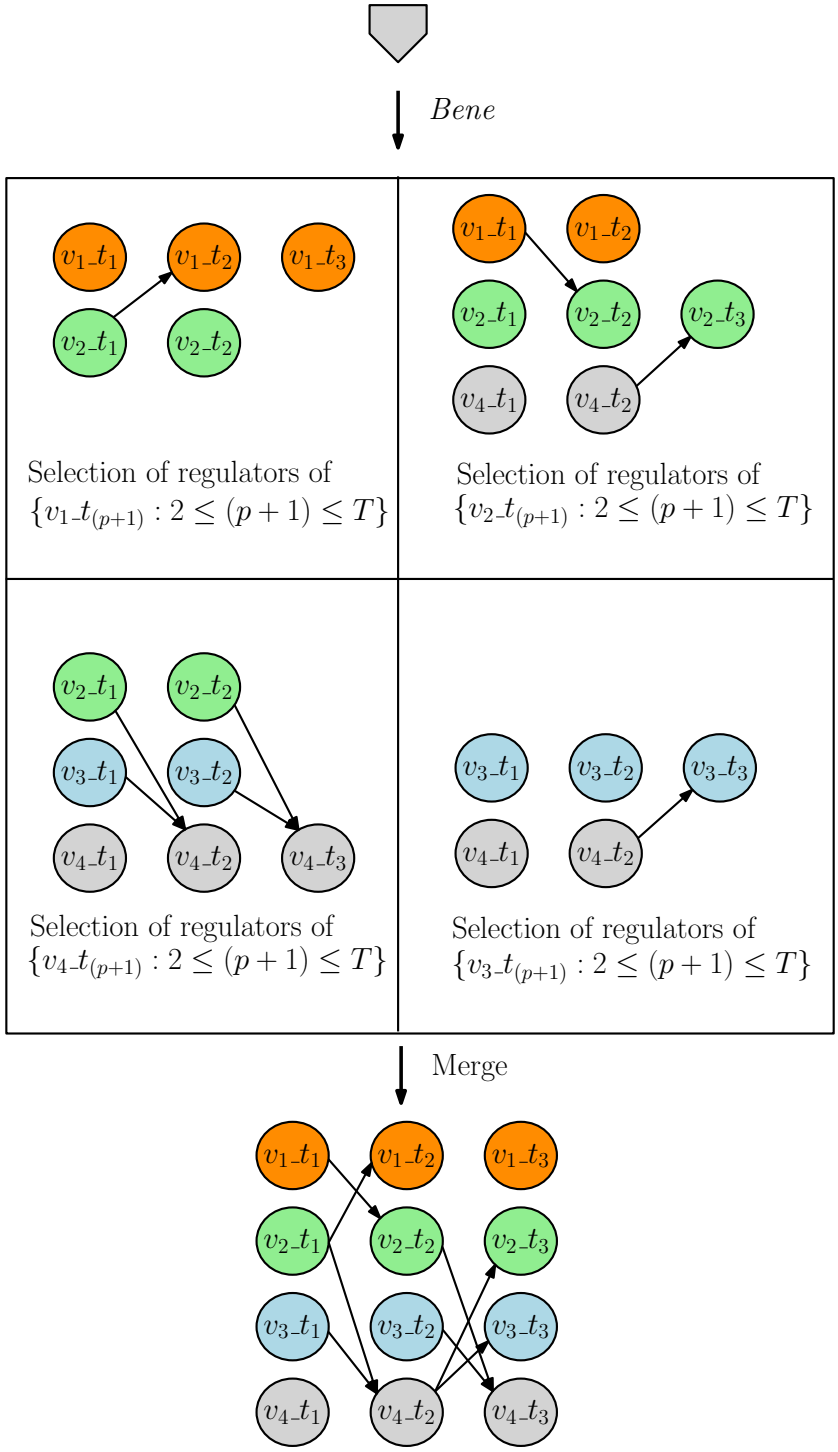


Figure 5.2: Graphical Flowchart (Part 2) of the *TGS* Algorithm. The flowchart is continued from Figure 5.1. For discussion of the *Bene* step, let us consider the 'Selection of regulators of $\{v_1\_t_{(p+1)} : 2 \leq (p+1) \leq T\}$'. Since, v_2 is the sole neighbour of v_1 in G_{CLR}, v_2 is the only candidate regulator of v_1 (Figure 5.1). Therefore, the candidate regulator sets of $v_1\_t_2$ are $\emptyset$ and $\{v_2\_t_1\}$. Among these two sets, *Bene* chooses $\{v_2\_t_1\}$ based on observations $\mathcal{D}_{\mathcal{V};\{t_1,t_2\};\mathcal{S}}$. Similarly, the candidate regulator sets of $v_1\_t_3$ are $\emptyset$ and $\{v_2\_t_2\}$. Among these two sets, *Bene* chooses $\emptyset$ based on observations $\mathcal{D}_{\mathcal{V};\{t_2,t_3\};\mathcal{S}}$.

$$
\begin{aligned}
T_{\mathrm{TGS}}(V) &= \left(\mathcal{O}\left(V^2\right) + o\left((T-1)\,V M_f^2\ 2^{(M_f-2)}\right)\right) && (5.1)\\
&= \left(\mathcal{O}\left(V^2\right) + o\left(o(V)\,V M_f^2\ 2^{(M_f-2)}\right)\right) && \because (T-1) = o(V)\\
&= \left(\mathcal{O}\left(V^2\right) + o\left(o(V)\,V\ o(\lg V)^2\ 2^{(o(\lg V)-2)}\right)\right) && \because M_f = o(\lg V)\\
&= \left(\mathcal{O}\left(V^2\right) + o\left(o\left(V^2\ (\lg V)^2\right)\frac{2^{(o(\lg V))}}{4}\right)\right)\\
&= \left(\mathcal{O}\left(V^2\right) + o\left(o\left(V^2\ (\lg V)^2\right)(o(V))\right)\right)\\
&= \left(\mathcal{O}\left(V^2\right) + o\left(V^3\ (\lg V)^2\right)\right)\\
&= o\left(V^3\ (\lg V)^2\right) && (5.2)
\end{aligned}
$$

5.2 Results

The results of the *TGS* algorithm on the benchmark datasets are presented in this section. *TGS*'s learning power and speed are evaluated against that of *TBN* and *ARTIVA*. *TBN* is included in this comparative study to analyse the effect of the *CLR* step.

5.2.1 Discretisation of the Datasets

The benchmark datasets are continuous in nature. However, *TBN* and *TGS* require discrete datasets. Hence, the benchmark datasets are discretised for *TBN* and *TGS*. For that purpose, the following two discretisation algorithms are utilised:

- **The *2L.wt* Algorithm:** It is based on the wild type (WT) values of the genes, provided through the DREAM challenge in Yeast1 steady state datasets. Expression of each gene is considered as a discrete random variable with two discrete levels $\{1, 2\}$. Each observed expression value (which is a real number) of that gene is converted to 1 if it is less the gene's WT value; else it is converted to 2.
- **The *2L.Tesla* Algorithm:** It is based on a domain-knowledge independent strategy (i.e. biological knowledge, like - WT values, are not used). The algorithm is described at (Ahmed and Xing, 2009, p. 2, Section 'Evolving Gene Networks During Drosophila melanogaster Development. Preprocessing the gene networks.', Supplementary Information).

5.2.2 Implementations

Both *TBN* and *TGS* are implemented in R programming language R Development Core Team (2008) version 3.3.2. For *CLR* and *Bene*, their implementations in R packages minet Meyer et al. (2008) (version: 3.34.0) and bnstruct Franzin et al. (2016) (version: 1.0.2) are used, respectively.

5.2.3 Learning From Dataset Ds10n

Dataset Ds10n is chosen over Ds10 for the comparison because: (a) it is noisy and hence more realistic than Ds10, and (b) Ds10n is used to evaluate algorithms in the DREAM

challenge while Ds10 is released after the challenge; therefore, the reader can compare the performances of the algorithms in this study with those of the algorithms employed during the challenge.

In the current study, *TBN* and *TGS* both perform better than *ARTIVA* in terms of learning power, except in FP (Table 5.1) [2]. They also outperform *ARTIVA* in speed (Table 5.2). Amongst *TBN* and *TGS*, it is found that *TGS* has faster learning speed. It is expected since the regulator search space for each gene, in case of *TGS*, is monotonically smaller than that of *TBN*. But the interesting observation is that *TGS*, being a heuristic based approximate search algorithm, performs competitively with *TBN*, an exhaustive search algorithm, in every metric of learning power as well. The reason behind that is explained by the fact that the *CLR* step in *TGS* captures 7 out of 10 true edges even from this noisy dataset; the high recall of the *CLR* step is utilised by the downstream *Bene* step to identify at least as many true edges identified by *TBN* while avoiding to search for as many potential false edges as possible. This reasoning is supported by another fact that *TGS* suffers from much less FP than *TBN*. Another observation is made that discretisation of input gene expression data based on domain-specific knowledge (as in wild type values of genes) improves learning compared to the domain-independent alternative (Table 5.1).

Table 5.1: Learning Power of the Selected Algorithms on Dataset Ds10n. TP = True Positive, FP = False Positive. Two ordered values in each cell for rows 'TBN' and 'TGS' represent application of two different data discretisation algorithms – *2L.wt* and *2L.Tesla*, respectively. On the other hand, other rows have a single value in each cell, since other algorithms do not require the dataset to be discretised. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced.

Algorithm	**TP**	**FP**	**Recall**	**Precision**	**F1**
TBN	(**3**, 1)	(17, 25)	(**0.3**, 0.1)	(0.15, 0.038)	(0.2, 0.056)
TGS	(**3**, 2)	(10, 12)	(**0.3**, 0.2)	(**0.231**, 0.143)	(**0.261**, 0.167)
ARTIVA	0	9	0	0	0
TVDBN-0	0	**1**	0	0	0
TVDBN-bino-hard	1	7	0.1	0.125	0.111
TVDBN-bino-soft	2	9	0.2	0.182	0.190

5.2.4 Learning From Datasets Ds50n and Ds100n

Due to *Bene*'s main memory requirement of $2^{(V+2)}$ Bytes (Silander and Myllymäki, 2006, Section 5), both *TBN* and *TGS* have the same inherent exponential memory requirement. In theory, that should enable them to learn a network with $V \leq 32$ with a 31 GB main memory, since $2^{(32+2)}$ Bytes = 16 GB < 31 GB. But it is found empirically that the bnstruct implementation of *Bene* can learn a network with $V \leq 15$ with that configuration, without any segmentation faults. Therefore, the max fan-in variant of *TGS* is employed for Ds50n and Ds100n with $M_f = 14$, since that would restrict each atomic network-learning problem to a maximum of 15 nodes (1 regulatee and a maximum of 14 candidate regulators). However, *TBN* does not have any such provisions and hence can not be applied on these datasets. As a result, *TBN* is excluded from the current study.

[2] Please note that algorithms *TVDBN-exp-hard* and *TVDBN-exp-soft* result in error for Ds10n.

Table 5.2: Runtime of the Selected Algorithms on Dataset Ds10n. Two ordered values in each cell for rows 'TBN' and 'TGS' represent application of two different data discretisation algorithms – *2L.wt* and *2L.Tesla*, respectively. On the other hand, other rows have a single value in each cell, since other algorithms do not require the dataset to be discretised. In *TGS*, the *CLR* step takes 0.003 seconds for *2L.wt* and *2L.Tesla* each.

Algorithm	**Ds10n**
TBN	(7.119s, 6.867s)
TGS	(5.789s, 5.76s)
ARTIVA	10m 20s
TVDBN-0	2m 24s
TVDBN-bino-hard	2m 15.2s
TVDBN-bino-soft	2m 14.6s

In this study, *ARTIVA* consistently outperforms *TGS* in FP, with considerable margins (Tables 5.3 and 5.4) [3]. Since the true GRNs are believed to be sparse in nature, it is expected that, among all possible regulatory relationships, only a few truly exist. For those larger number of relationships, that do not exist, *ARTIVA* is less likely than *TGS* to mistake them as true relationships. However, *ARTIVA* tends to over-estimate the non-existent relationships by mistaking a large number of true relationships as non-existent, as evident from its considerably lower TP compared to those of *TGS*. Another major concern with *ARTIVA* is the runtime. It takes almost 32 hours to reconstruct 100-gene GRNs, which is certainly a bottleneck for its application in reconstructing human genome-scale GRNs (Table 5.5). In comparison, *TGS* consumes only about 18 minutes. Moreover, *TGS*'s runtime grows almost linearly as the number of genes grow (Figure 5.3). These observations indicate that *TGS* is substantially more suitable for reconstructing large-scale GRNs than *ARTIVA*.

Table 5.3: Learning Power of the Selected Algorithms on Dataset Ds50n. TP = True Positive, FP = False Positive. Algorithm *2L.wt* is used for data discretisation in *TGS*. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced.

Algorithm	**TP**	**FP**	**Recall**	**Precision**	**F1**
TGS	**15**	342	**0.195**	0.042	0.069
ARTIVA	6	**64**	0.078	**0.086**	**0.082**
TVDBN-0	7	199	0.091	0.034	0.049
TVDBN-bino-hard	11	410	0.143	0.026	0.044
TVDBN-bino-soft	14	395	0.182	0.034	0.058

[3]Please note that algorithms *TVDBN-exp-hard* and *TVDBN-exp-soft* result in error for Ds50n and Ds100n.

Table 5.4: Learning Power of the Selected Algorithms on Dataset Ds100n. TP = True Positive, FP = False Positive. Algorithm *2L.wt* is used for data discretisation in *TGS*. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced.

Algorithm	**TP**	**FP**	**Recall**	**Precision**	**F1**
TGS	**28**	790	**0.169**	0.034	0.057
ARTIVA	14	**158**	0.084	**0.081**	**0.083**
TVDBN-0	9	678	0.054	0.013	0.021
TVDBN-bino-hard	26	1304	0.157	0.020	0.035
TVDBN-bino-soft	18	1296	0.108	0.014	0.024

Table 5.5: Runtime of the Selected Algorithms on Datasets Ds50n and Ds100n. For *TGS*, algorithm *2L.wt* is used for data discretisation. The *CLR* step in *TGS* takes 0.005 and 0.013 seconds for Ds50n and Ds100n, respectively.

Algorithm	**Ds50n**	**Ds100n**
TGS	7m 36s	17m 49s
ARTIVA	4h 30m 15s	31h 52m 54s
TVDBN-0	11m 59s	52m 17s
TVDBN-bino-hard	9m 38s	2h 53m 32s
TVDBN-bino-soft	8m 8s	17m 20s

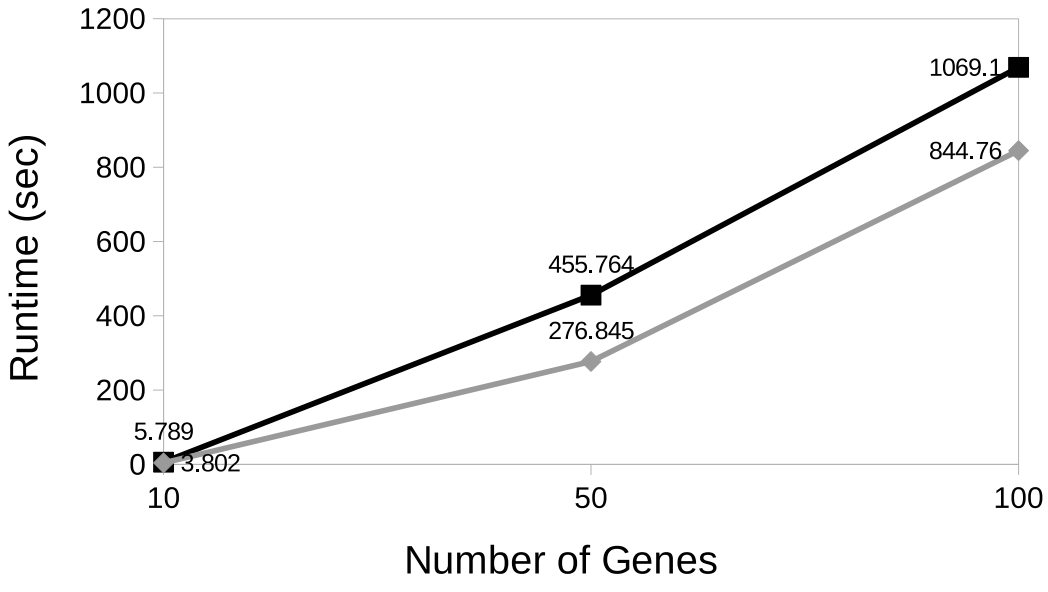


Figure 5.3: Runtime of the *TGS* Algorithm w.r.t. the Number of Genes in the Benchmark Datasets (Table 4.1). The black and grey lines represent noisy and noiseless versions of the datasets, respectively.

5.2.5 Effects of Noise on Learning Power and Speed

TGS is evaluated on all noisy and noiseless datasets with different number of genes. From Figures 5.3 and 5.4 , it can be observed that the presence of noise negatively impacts runtime and precision. This observation can be explained by analysing the effect of noise on the *CLR* step (Table 5.6). In the absence of noise, the *CLR* step can eliminate more numbers of potential false regulators from the candidate set of regulators of each regulatee, resulting in smaller and more precise shortlist of candidate regulators. That in turn, improves precision and speed of the overall algorithm.

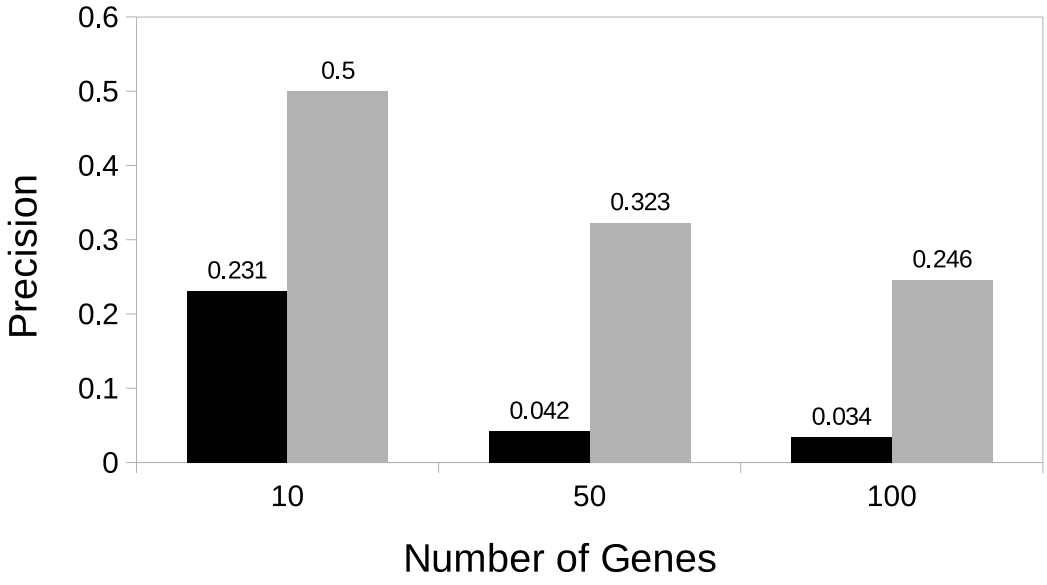


Figure 5.4: Precision of the *TGS* Algorithm w.r.t. the Number of Genes in the Benchmark Datasets. The black and grey bars represent noisy and noiseless versions of the datasets, respectively. The *2L.wt* algorithm is used for data discretisation.

Table 5.6: Maximum Number of Neighbours a Gene has in the *CLR* Network. Algorithm *2L.wt* is used for data discretisation.

Total Number of Genes	***Noiseless Dataset***	***Noisy Dataset***
10	4	7
50	24	33
100	43	84

5.3 Excerpt and Future Work

In this chapter, a novel algorithm, namely *TGS*, is proposed to reconstruct time-varying GRNs from time-series gene expression datasets. *TGS* assumes that there are multiple time series in the dataset. Additionally, it assumes that there are no missing values.

TGS employs a two-step learning framework. In the first step, for each target gene, a shortlist of its candidate regulators is inferred. In the final step, these shortlisted can-

didates are thoroughly examined to identify the true regulators. Moreover, the temporal sequence of the regulatory events is learnt from the data.

The novelty of the *TGS* algorithm is two-fold: (A) flexibility and (B) time-efficiency. Its flexible framework allows time-varying GRNs to be learnt independently of each other. *TGS* learns every GRN structure in a data-driven manner, without imposing the smoothly time-varying assumption.

However, *ARTIVA* is already able to provide a similarly flexible framework (Lèbre et al., 2010, Section 'Conclusions'). The only challenge with *ARTIVA* is its substantial runtime. That makes *ARTIVA*'s application prohibitive with large-scale datasets. *TGS*, on the other hand, is able to offer the same flexibility but in a significantly more time-efficient manner. It requires only around 29 minutes for a microarray dataset with 4028 genes (and 15 time series each comprised of 2 time points) (Pyne et al., 2020).

It needs to be noted that, another existing algorithm, namely *TV-DBN*, is also able to demonstrate its scalability to a microarray dataset with 3626 genes (runtime is unknown) (Song et al., 2009b). Nevertheless, *TV-DBN* accomplishes such scalability by imposing the smoothly time-varying assumption. The assumption states that each GRN shares more common edges with its temporally adjacent GRNs than with the distal ones.

On the other hand, *TGS*'s framework is compatible with any dataset regardless of whether the smoothly time-varying assumption holds for it or not. Moreover, *TGS* consistently outperforms *ARTIVA* in true positive detection.

Nevertheless, there are scopes for improvement. One limitation of *TGS* is the need to discretise continuous datasets. One reason behind that is *TGS* uses BIC score to determine the best GRN structure and BIC scoring function requires the data to be discretised. Two of the ways the issue can be resolved are: (a) by using a scoring function that does not require the input data to be discretised, like in Grzegorczyk and Husmeier (2009), and (b) by developing a regression-based structure learning strategy, like- Lèbre et al. (2010), since regression problems are inherently compatible with continuous data.

However, the experiments with the large datasets help us to identify the Achilles' heel of *TGS*. It is the fact that its main memory requirement grows exponentially with the number of genes (and in turn number of candidate regulators for each gene) in a given dataset. In the current implementation of *TGS*, the maximum number of candidate regulators is restricted to fourteen for each gene, to avoid this issue. Relaxing this restriction is an important challenge since the true number of regulators for a gene is not known a priori.

The reason behind such astronomical memory requirement is that *Bene* and the related Bayesian Network structure learning algorithms need to compute and store the global conditional probability table (Silander and Myllymäki, 2006, Section 3.1) in main memory. Some researchers are exploring efficient ways to distribute this task and storage across multiple computing nodes using distributed computing strategies, e.g., Jahnsson et al. Jahnsson et al. (2017). Mending this gap can be considered a worthwhile challenge. The convergence of affordable high-throughput gene expression measurement technologies with accurate and scalable GRN reconstruction methodologies will be a valuable achievement. It will help in improving our understanding of disease progression and life in general through the lens of gene regulation.

Lastly, we note the immediate challenge that *TGS* is facing. Although *TGS* outperforms *ARTIVA* in recall, the former is unable to compete with the latter in precision. As a result, *ARTIVA* retains its superiority in F1-score. Therefore, the immediate challenge in this thesis is to develop algorithms that are competitive to *TGS* in time-efficiency

and competitive to *ARTIVA* in F1-score.

5.4 Contributions

The TGS algorithm is jointly designed by Mr Alok Ranjan Kumar, Dr Ashish Anand and the author. The implementations and evaluations of the same are conducted by the author himself. This work is published in IEEE/ACM Transactions on Computational Biology and Bioinformatics (Pyne et al., 2020).

5.5 Chapter Summary

In the previous chapter, we formulate the objective of this thesis (Chapter 4). The objective is to develop reconstruction algorithms that can deliver correctness competitive to that of *ARTIVA* and computational efficiency compatible with large-scale datasets. In this chapter, we propose an algorithm named *TGS* which offers time-efficiency compatible with large-scale datasets. However, it does not deliver correctness competitive to that of *ARTIVA*. Moreover, *TGS*'s memory-efficiency is not compatible with large-scale datasets.

Chapter 6

Balancing Recall and Precision

In the previous chapter, we propose a reconstruction algorithm, namely *TGS*, that offers time-efficiency compatible with large-scale datasets having hundreds to thousands of genes (Chapter 5). However, *TGS* meets the thesis objective only partially, for the following two reasons:

- *TGS* is unable to provide correctness competitive to that of *ARTIVA*; and
- *TGS* does not offer memory-efficiency compatible with large-scale datasets.

In this chapter, we focus on improving correctness. Although *TGS* outperforms *ARTIVA* in recall, the former can not compete with the latter in precision. For this reason, *TGS* is unable to provide correctness competitive to that of *ARTIVA*. Hence, we propose another algorithm, namely 'TGS-Plus' (TGS+). This algorithm offers recall competitive to that of *TGS* and precision competitive to that of *ARTIVA*. As a consequence, *TGS+* supersedes *ARTIVA* in F1-score. Moreover, *TGS+* offers time-efficiency compatible with large-scale datasets.

The chapter is organised into multiple sections. In Section 6.1 , we design a novel algorithm. Subsequently, the experimental results are discussed in Section 6.2 . An excerpt along with a pointer to the future work are provided in Section 6.3 . The research contributions are acknowledged in Section 6.4 . Finally, in Section 6.5 , a summary of the chapter is presented.

6.1 Methods

In this section, we propose a variant of the *TGS* algorithm named 'TGS-Plus' (*TGS+*). It differs from *TGS* only in the *CLR* step. In *TGS+* (Algorithm 13), the *CLR* step does not compute the *CLR* weights from the raw mutual information matrix, as computed in *TGS* (Algorithm 10). Instead it feeds the raw mutual information matrix to the *ARACNE* algorithm (Section 3.1.3.2), which refines the matrix. This refined matrix is then used for the calculation of the *CLR* weights. The reason for applying *ARACNE* is to reduce the false positive mutual informations to zero. A pair of genes, which do not possess a regulatory relationship, can still have a positive mutual information if their expressions are correlated (e.g., both of them are regulated by the same regulators). Such false positive mutual informations are detected by *ARACNE* and their mutual information is reduced to zero in the mutual information matrix. However, *ARACNE* may underestimate some true positive mutual informations and reduce them to zero as well. Due to that reason, it is expected that *TGS+* will produce considerably less

numbers of false positives than those of *TGS*, while also producing less numbers of true positives as a trade-off.

This trade-off between true positives and false positives can be very useful for the users. A user, who wishes to experimentally verify the predicted edges, one by one, would prefer to have lower false positive edges, even at the cost of lower true positives. The reason is that each false positive edge leads to an unnecessary set of experiments, causing wastage of valuable resources. On the other hand, a user, who wishes to apply other computational methods on the predicted network, would prefer to have as many true positive edges as possible for further processing, even at the cost of a higher number of false positives. Thus, two variants of the *TGS* algorithm cater to two different sets of demands from the users.

6.2 Results

The results of the *TGS+* algorithm on the benchmark datasets are presented in this section. *TGS*'s learning power and speed are evaluated against that of *TGS* and *ARTIVA*. *TGS* is included in this comparative study to analyse the effect of the mutual-information refinement strategy.

6.2.1 Implementations

TGS+ is implemented in R programming language R Development Core Team (2008) version 3.3.2. For *CLR* and *Bene*, their implementations in R packages minet Meyer et al. (2008) (version: 3.34.0) and bnstruct Franzin et al. (2016) (version: 1.0.2) are used, respectively.

6.2.2 Learning from the Benchmark Datasets

In this section, we comparatively study the performance of *TGS+* against its alternatives on datasets Ds10n, Ds50n and Ds100n. Since *TGS+* inherits the memory-inefficiencies of *TGS*, we use a max fan-in value of 14 for both of them.

As expected, the false positives are considerably reduced in *TGS+*, compared to that of *TGS* (Tables 6.1, 6.2 and 6.3) [1]. As also expected, the trade-off is made with the true positives which are reduced as well. However, the true positives in *TGS+* still remain monotonically higher than those in *ARTIVA*. As a result, *TGS+* supersedes *ARTIVA* in F1-score for two of the three datasets.

An interesting observation is made with runtime. The runtime decreases drastically in *TGS+*, compared to that in *TGS* (Tables 6.4). This is a positive side-effect of the *ARACNE* step. *ARACNE* produces a considerably sparser (less non-zero values) mutual information matrix than the raw mutual information matrix. In turn, the *CLR* step produces a sparser *CLR* network. Therefore, most of the regulatee genes have much lower number of candidate regulators to be examined in the *BN* step. Thus, the latter step consumes considerably less amount of time, which is reflected in *TGS+*'s runtime. For instance, *TGS+* takes only 1 minute for Ds100n, whereas *ARTIVA* and *TGS* consume 32 hours and 18 minutes, respectively (Table 6.4).

[1] Please note that algorithms *TVDBN-exp-hard* and *TVDBN-exp-soft* result in error for all datasets.

Table 6.1: Learning Power of the Selected Algorithms on Dataset Ds10n. TP = True Positive, FP = False Positive. Algorithm *2L.wt* is used for data discretisation in case of *TGS* and *TGS+*. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced.

Algorithm	**TP**	**FP**	**Recall**	**Precision**	**F1**
TGS	**3**	10	**0.3**	0.231	0.261
TGS+	**3**	1	**0.3**	**0.75**	**0.429**
ARTIVA	0	9	0	0	0
TVDBN-0	0	1	0	0	0
TVDBN-bino-hard	1	7	0.1	0.125	0.111
TVDBN-bino-soft	2	9	0.2	0.182	0.190

Table 6.2: Learning Power of the Selected Algorithms on Dataset Ds50n. TP = True Positive, FP = False Positive. Algorithm *2L.wt* is used for data discretisation in case of *TGS* and *TGS+*. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced.

Algorithm	**TP**	**FP**	**Recall**	**Precision**	**F1**
TGS	**15**	342	**0.195**	0.042	0.069
TGS+	6	100	0.078	0.057	0.066
ARTIVA	6	**64**	0.078	**0.086**	**0.082**
TVDBN-0	7	199	0.091	0.034	0.049
TVDBN-bino-hard	11	410	0.143	0.026	0.044
TVDBN-bino-soft	14	395	0.182	0.034	0.058

Table 6.3: Learning Power of the Selected Algorithms on Dataset Ds100n. TP = True Positive, FP = False Positive. Algorithm *2L.wt* is used for data discretisation in case of *TGS* and *TGS+*. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced.

Algorithm	**TP**	**FP**	**Recall**	**Precision**	**F1**
TGS	**28**	790	**0.169**	0.034	0.057
TGS+	19	181	0.114	**0.095**	**0.104**
ARTIVA	14	**158**	0.084	0.081	0.083
TVDBN-0	9	678	0.054	0.013	0.021
TVDBN-bino-hard	26	1304	0.157	0.020	0.035
TVDBN-bino-soft	18	1296	0.108	0.014	0.024

6.3 Excerpt and Future Work

In this chapter, a novel algorithm, namely *TGS+*, is proposed to reconstruct time-varying GRNs from time-series gene expression datasets. *TGS+* assumes that there are multiple time series in the dataset. Additionally, it assumes that there are no missing

Table 6.4: Runtime of the Selected Algorithms on the Benchmark Datasets. Algorithm *2L.wt* is used for data discretisation in case of *TGS* and *TGS+*.

Algorithm	**Ds10n**	**Ds50n**	**Ds100n**
TGS	5.789s	7m 36s	17m 49s
TGS+	5.515s	22.034s	1m 4s
ARTIVA	10m 20s	4h 30m 15s	31h 52m 54s
TVDBN-0	2m 24s	11m 59s	52m 17s
TVDBN-bino-hard	2m 15.2s	9m 38s	2h 53m 32s
TVDBN-bino-soft	2m 14.6s	8m 8s	17m 20s

values.

TGS+ follows the same two-step framework as *TGS*. In the first step, a shortlist of candidate regulators is prepared for every regulatee gene. In the second step, the shortlist is thoroughly examined to select the final set of regulators. *TGS+* differs from *TGS* in the way short-listing is performed. In *TGS*, short-listing is performed based on the 'raw' mutual information matrix estimated from the dataset. However, in *TGS+*, the raw matrix is refined through an algorithm named *ARACNE*. This algorithm is well-known for removing a significant number of false-positive mutual information values at the cost of a reasonable number of true-positive ones. Subsequently, the 'refined' mutual information matrix is used for the short-listing task.

It is empirically observed that *TGS+* incurs considerably less number of false-positive edges than that of *TGS*. Thus, *TGS+* achieves significantly higher precisions than that of *TGS* and competitive to that of *ARTIVA*. On the other hand, the former obtains competitive recalls, compared to the latter. As a consequence, *TGS+* supersedes *ARTIVA* in F1-score.

At the same time, *TGS+* overtakes *TGS* and *ARTIVA* in runtime. The former processes the 100-gene dataset in only a minute. The same dataset takes *ARTIVA* and *TGS* around 1.5 days and 18 minutes, respectively.

Nevertheless, *TGS+* inherits the same memory-inefficiencies as *TGS*. Its memory requirement grows exponentially with the number of candidate regulators in a shortlist. Therefore, *TGS+* may not be applicable for large-scale datasets where shortlists can contain hundreds of genes.

6.4 Contributions

This work is published in IEEE/ACM Transactions on Computational Biology and Bioinformatics (Pyne et al., 2020).

6.5 Chapter Summary

In the previous chapter, we propose a reconstruction algorithm named *TGS* (Chapter 5). This algorithm offers time-efficiency compatible with large-scale datasets. However, it does not offer correctness competitive to that of *ARTIVA*. In this chapter, we propose another algorithm named *TGS+* which offers time-efficiency compatible with large-scale datasets and correctness competitive to that of *ARTIVA*. However, *TGS+*'s memory-

efficiency is not compatible with large-scale datasets.

Chapter 7

Improving Memory-efficiency

In the previous chapters, we propose two novel algorithms for the task of reconstructing time-varying gene regulatory networks from a given time-series gene expression dataset. The first algorithm, namely *TGS*, offers time-efficiency compatible with large-scale datasets (Chapter 5). The second algorithm, namely *TGS+*, offers competitive time-efficiency to that of *TGS* as well as correctness competitive to that of *ARTIVA* (Chapter 6). However, both the proposed algorithms suffer from memory-inefficiencies. Their memory requirements increase exponentially with the number of genes in the given dataset. This issue makes them incompatible with large-scale datasets. In this chapter, we mitigate this issue by designing a third set of algorithms that are more memory-efficient.

The chapter is organised into multiple sections. In Section 7.1 , we recapitulate the thesis objective. In Section 7.2 , we measure our progress made in the previous chapters and discuss its limitations. In Section 7.3 , we investigate the origin of the aforementioned limitations. In Section 7.4 , we present a novel idea for overcoming these limitations. Based on the novel idea, we propose two novel algorithms in Section 7.5 . Section 7.6 discusses the experimental results of the proposed algorithms. An excerpt along with a pointer to the future work are provided in Section 7.7 . The research contributions are acknowledged in Section 7.8 . Finally, in Section 7.9 , a summary of the chapter is presented.

7.1 Task at Hand: Revisited

To preserve the continuity of this chapter, we briefly revisit the task at hand i.e. the problem statement in this section. The problem is related to Gene Regulatory Networks (GRNs); a GRN is a directed network where nodes represent genes; a directed edge from a gene v_i to a gene v_j implies that the former gene is a 'regulator' of the latter gene (the 'regulatee').

However, these regulator-regulatee relationships are time-varying; it means that not all the regulators of a particular gene have regulatory effects on the latter gene all the time. Therefore, a fundamental question in the Systems Biology is 'who regulates whom and when?'

The reason the aforementioned question is considered as fundamental in the Systems Biology is that a large number of biological processes are temporal in nature, such as developmental programs (how an organism grows from embryo to adulthood) and pathogenesis (how a disease develops and progresses inside an organism). If we are able to identify how the edges (hereafter, 'structure') of the GRN underlying such a process

change with time, we might be able to understand the process at a molecular level.

However, hitherto there are no publicly available technologies that allow us to visualise the structural changes in a GRN. Therefore, a widely-used approach is to follow a two-step pipeline. In the first step, the 'expressions' of the concerned genes are measured at specific time intervals over a pre-decided period of time.

The expression of a gene at a particular time point is usually measured by the quantity of mRNAs corresponding to that gene present in a single unit of sample, e.g., one ml of blood, drawn at that time point. Therefore, an expression value is generally a non-negative real number. Having measured the expression values of all concerned genes at all the time points results in a time-series gene expression dataset.

Given the aforementioned dataset, the fluctuations in the expression values of each gene over time is dependent on the structural changes in the underlying GRN. Therefore, the second step is to reverse-engineer (hereafter, 'reconstruct') the time-varying GRN structures from the given dataset. This particular step is known as the 'time-varying GRN reconstruction from time-series gene expression data'.

When the given dataset is high-dimensional i.e. it contains hundreds or thousands of genes, the task becomes infeasible to accomplish completely manually. In that case, computational algorithms are employed to accomplish this task. Designing an algorithm that can conduct the reconstruction task correctly and efficiently is a long-standing challenge in the Computational Systems Biology.

This PhD thesis is conceived to make a stride towards overcoming this challenge. The challenge is formally defined in Chapter 4 . Hence, we conclude this section by summarising the notations that will be useful for this chapter:

Desired Input: A time-varying GRN reconstruction algorithm takes a dataset $\mathcal{D}$ as input (Figure 7.1). Dataset $\mathcal{D}$ consists of a set of time series $\mathcal{S} = \{s_1, \ldots, s_S\}$. Each time series is comprised of the expression levels a set of genes $\mathcal{V} = \{v_1, \ldots, v_V\}$ at T consecutive time points $\mathcal{T} = \{t_1, \ldots, t_T\}$. $\mathcal{D}_{(\mathcal{X};\mathcal{Y};\mathcal{Z})}$ denotes the expression levels of genes $\mathcal{X}$ at time points $\mathcal{Y}$ in time series $\mathcal{Z}$. Therefore, $\mathcal{D}_{(\mathcal{X};\mathcal{Y};\mathcal{Z})} \subseteq \mathcal{D}$, where $\mathcal{X} \subseteq \mathcal{V}$, $\mathcal{Y} \subseteq \mathcal{T}$, $\mathcal{Z} \subseteq \mathcal{S}$. $\mathcal{D}$ is assumed to be 'complete' i.e. $\mathcal{D}$ does not have any missing values.

Desired Output: Given dataset $\mathcal{D}$, the objective of the algorithm is to return a temporally ordered sequence of GRNs as output. The output sequence is denoted by $\mathcal{G} = \left(G^{(1)}, \ldots, G^{(T-1)}\right)$. Each $G^{(p)} (\in \mathcal{G})$ is a time-interval-specific GRN; it represents the gene regulatory events occurred during the time interval between time points t_p and $t_{(p+1)}$. Structurally, $G^{(p)}$ is a directed unweighted network with $(2 \times V)$ nodes: $\{v_i\_t_q \colon v_i \in \mathcal{V}, t_q \in \{t_p, t_{(p+1)}\}\}$ (Figure 7.1). Each node $v_i\_t_q$ is a distinct random variable, which represents the expression level of gene v_i at time point t_q; hence, data points $\mathcal{D}_{(\{v_i\};\{t_q\};\mathcal{S})}$ are considered to be S instances of random variable $v_i\_t_q$. It is assumed that the underlying gene regulation process is first order Markovian i.e. node $v_i\_t_q$ can have regulatory effects only on the nodes at the immediately next time point $t_{(q+1)}$, if any (Friedman et al., 1998). Thus, there exists a directed edge $\left(v_i\_t_q, v_j\_t_{(q+1)}\right)$ in $\mathcal{G}$ if and only if $v_i\_t_q$ has a regulatory effect on $v_j\_t_{(q+1)}$, implying that the expression level of gene v_i at time point t_q plays a regulatory role on that of gene v_j at time point $t_{(q+1)}$.

7.2 Limitations of the Previously Proposed Algorithms

In the previous chapters, we propose two algorithms for the task at hand. In this section, we discuss their limitations, overcoming which is the motivation behind this chapter.

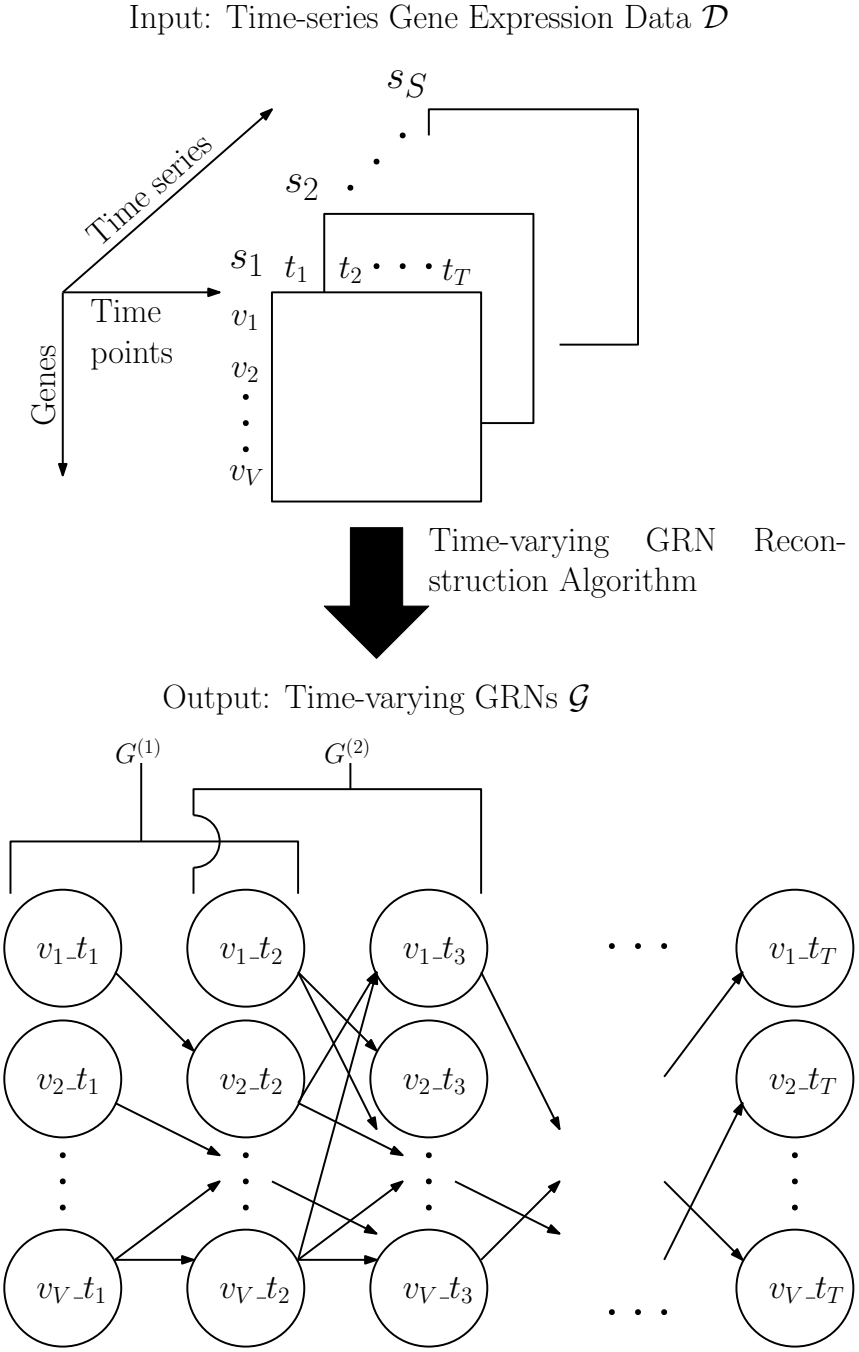


Figure 7.1: The Workflow of a Time-varying GRN Reconstruction Algorithm. The algorithm takes a time-series gene expression data $\mathcal{D}$ as input. The data consists of S number of time series. Each time series contains measured expressions of V number of genes across T number of time points. In return, the algorithm outputs time-varying GRNs $\left(G^{(1)}, \ldots, G^{(T-1)}\right) = \mathcal{G}$, which is a sequence of directed unweighted networks. Here, $G^{(p)} (\in \mathcal{G})$ represents the gene regulatory events occurred during the time interval between time points t_p and $t_{(p+1)}$. It consists of $(2 \times V)$ nodes $\{v_i\_t_q \colon v_i \in \mathcal{V}, t_q \in \{t_p, t_{(p+1)}\}\}$. There exists a directed unweighted edge $\left(v_i\_t_p, v_j\_t_{(p+1)}\right)$ if and only if v_i regulates v_j during time interval $\left(t_p, t_{(p+1)}\right)$. For instance, $G^{(1)}$ represents the regulatory events that occurred between time points t_1 and t_2. One such event is the regulatory effect of $v_1\_t_1$ on $v_2\_t_2$ as represented by a directed edge.

The names of the previously proposed algorithms are *TGS* and *TGS+*. Both the algorithms divide the task into smaller atomic problems. Each atomic problem is concerned with finding the regulators of a distinct node in $\mathcal{G}$ except the nodes in the first time point. To solve that problem, both *TGS* and *TGS+* prepare a shortlist of candidate regulators for the concerned node. Subsequently, they select the best set of regulators from the shortlist. For selecting the best set, they employ a structure-learning algorithm named Bene (Silander and Myllymäki, 2006). Hence, we address the final selection step as the 'Bene step'. In this step, the main memory requirement (hereafter, simply 'memory requirement') increases exponentially with the number of candidate regulators. As Jahnsson et al. aptly put it,"The algorithm can always be given more time; however, if it exceeds the available memory resources, nothing can be done to solve the instance" (Jahnsson et al., 2017).

To mitigate the memory-efficiency of the Bene step, *TGS* and *TGS+* use a user-defined parameter named 'max fan-in' [1]. This parameter accepts a positive integer value from the user and restricts the maximum number of candidate regulators to that value. For example, if 'max fan-in = 14', then at most fourteen candidate regulators can be shortlisted for each node. However, relaxing this restriction is necessary since the actual number of regulators for a node is not known a priori.

7.3 Investigations into the Origin of the Limitations

In this section, we investigate the reason behind the memory-inefficiency of the Bene step. Given the shortlist of candidate regulators prepared in the previous step, the Bene step selects the best set of regulators from the shortlist. Let us denote the given shortlist of candidate regulators for node $v_j\_t_{(p+1)}$ as $\mathcal{V}_{(j;(p+1))}$. If the value of the max fan-in parameter is M_f, then the size of this shortlist is $|\mathcal{V}_{(j;(p+1))}| \leq M_f = \mathcal{O}(M_f)$. Given shortlist $\mathcal{V}_{(j;(p+1))}$, the Bene step finds its highest scoring subset with respect to a scoring function known as Bayesian Information Criterion (BIC) (Markowetz and Spang, 2007). For that purpose, the Bene step needs to calculate the BIC scores of all subsets of $\mathcal{V}_{(j;(p+1))}$. Since $\mathcal{V}_{(j;(p+1))} = \mathcal{O}(M_f)$, there are a total of $2^{|\mathcal{V}_{(j;(p+1))}|} = 2^{\mathcal{O}(M_f)}$ subsets. For each subset, the score is calculated. During score calculations, the Bene step requires all $2^{\mathcal{O}(M_f)}$ subsets and their corresponding scores to be simultaneously held in memory. For the aforementioned reason, the memory requirement of the Bene step increases exponentially with the number of candidate regulators.

7.4 A Novel Idea for Overcoming the Limitations

In this section, we present a novel idea for accomplishing the goal of the Bene step with significantly less memory. As discussed in the previous section, the Bene step requires $2^{|\mathcal{V}_{(j;(p+1))}|}$ subsets and $2^{|\mathcal{V}_{(j;(p+1))}|}$ scores to be stored in memory. It does so for finding the subset with the highest score. However, the same goal can be achieved with $2^{|\mathcal{V}_{(j;(p+1))}|}$ subsets and only two scores along with two pointers; the idea is illustrated in Figure 7.2 . It can be noted that the required gene expression data also needs to be stored in memory for score calculations. Since the required data is same as that of the Bene step, we can omit its discussion from the comparative analysis.

[1] 'Fan-in' is a number specific to each node in a network; it represents the number of regulators the node has.

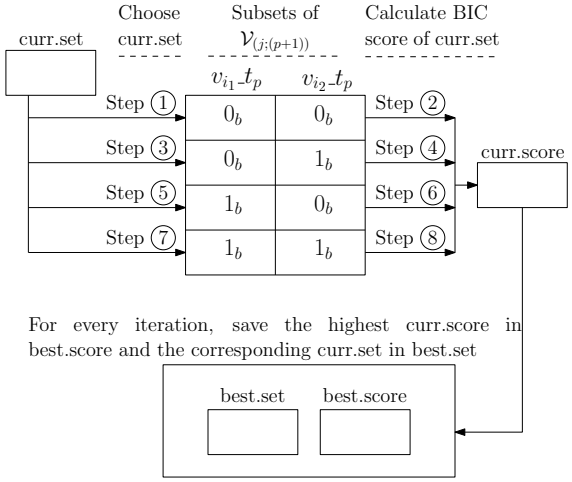


Figure 7.2: Illustration of the Idea for Finding the Highest Scoring Subset with $2^{|\mathcal{V}_{(j;(p+1))}|}$ Subsets, Two Scores and Two Pointers. Each rectangle represents a location in the memory. It is assumed that $\mathcal{V}_{(j;(p+1))} = \{v_{i_1}\_t_p, v_{i_2}\_t_p\}$. Therefore, its subsets are $\{\emptyset, \{v_{i_2}\_t_p\}, \{v_{i_1}\_t_p\}, \{v_{i_1}\_t_p, v_{i_2}\_t_p\}\}$. Each subset is represented as a binary string of Boolean TRUE (1_b) and FALSE (0_b) values, indicating whether a gene is present or not in that subset, respectively. All the subsets are held in memory simultaneously as depicted by the four-row two-column table in the middle. In addition, two pointers ('curr.set' and 'best.set') and two scores ('curr.score' and 'best.score') are also stored in memory (each score is a floating point number). In step 1, the first subset is pointed to by curr.set (current subset). In step 2, the score of this subset is calculated and stored inside curr.score (current subset's score). Subsequently, the values of curr.set and curr.score are copied to best.set (hitherto best subset) and best.score (hitherto best score), respectively. In step 3, the second subset is pointed to by curr.set. In step 4, its score is calculated and stored inside curr.score. If curr.score is greater than best.score, then the values of best.score and best.set are replaced with that of curr.score and curr.set, respectively. Otherwise, if curr.score is less than or equal to best.score, then best.score and best.set are kept unchanged. Next in step 5, the third subset is pointed to by curr.set and the same procedure is followed till all the subsets are exhausted. As a result, the final value of best.set points to the subset with the highest score (ties between two equally scoring subsets are broken in favour of the subset with the lower index). Overall, this strategy requires two pointers (curr.set and best.set), two scores (curr.score and best.score) and $2^{|\mathcal{V}_{(j;(p+1))}|} = 2^2 = 4$ subsets to be held in memory. It also requires corresponding gene expression data (not shown in figure) to be stored in memory for calculating the scores.

Nevertheless, the number of subsets in memory remains exponential to $|\mathcal{V}_{(j;(p+1))}|$. One naive way to reduce this number is to retain only the subsets pointed to by curr.set and best.set, and move the rest of the subsets to the secondary storage (hereafter, 'disk'). However, that will require copying each subset from disk to memory when that subset's score needs to be calculated. Therefore, this strategy will increase the runtime significantly due to costly disk Input/Outputs (I/Os).

There is one way to avoid costly disk I/Os. Instead of generating all the subsets at once and storing them in disk, we can generate a subset immediately before its score needs to be calculated. If we are able to do that, we only need to store two subsets – current subset and hitherto best subset – in memory at any point of time.

We find an idea to generate a subset in real-time. The idea stems from the observa-

tion that every non-empty subset can be generated by adding 1_b to the Least Significant Bit of the previous subset (Figure 7.2). The idea is illustrated in Figure 7.3 .

7.5 Design of Novel Algorithms Based on the Novel Idea

In this section, we design two novel algorithms based on the novel idea proposed in the previous section. First, we design an algorithm named 'TGS - which is Light on memory' (TGS-Lite) (Algorithm 16). This algorithm is equivalent to the *TGS* algorithm except for the Bene step. In *TGS-Lite*, the Bene step is replaced with the novel strategy ideated in the previous section. Second, we design another algorithm named 'TGS-Lite-plus' (TGS-Lite+) (Algorithm 17) by replacing the Bene step in the *TGS+* algorithm with the same novel strategy.

TGS-Lite preserves the state-of-the-art time complexity of *TGS* which is $o\big(V^2 \lg V\big)$ (Pyne and Anand, 2019a). Since there are no differences between the ways *TGS-Lite* and *TGS* score and compare candidate subsets, no dissimilarities are expected between their correctnesses. The only difference should be in their memory-efficiencies. In that regard, *TGS-Lite* is expected to consume significantly less amount of memory than that of *TGS*.

Similarly, *TGS-Lite+* maintains the time complexity of *TGS+* which is $\mathcal{O}(V^3)$ (Pyne and Anand, 2019a). *TGS-Lite+* is also expected to have the same correctness as *TGS+* at the benefit of a significantly lower memory requirement.

7.6 Experimental Results

In this section, we present the experimental results of the novel algorithms proposed in the previous section. For the experiments, we utilise benchmark datasets Ds10n, Ds50n and Ds100n. Dataset Ds10n contains 4 time series, each of which is comprised of the expression levels of 10 genes across 21 consecutive time points. Dataset Ds50n, on the other hand, contains 23 time series, each of which is comprised of the expression levels of 50 genes across 21 consecutive time points. Lastly, dataset Ds100n contains 46 time series, each of which is comprised of the expression levels of 100 genes across 21 consecutive time points.

Although, the datasets are longitudinal, the true networks available with the datasets are time-invariant. Thus, each dataset has a true summary GRN. On the other hand, both *TGS-Lite* and *TGS-Lite+* output a sequence of GRNs, length of the sequence being one less than the number of time points. To make the output comparable with the true GRN, we 'roll' the output into a single GRN using the same strategy as used for *TGS* and *TGS+*.

Like *TGS* and *TGS+*, *TGS-Lite* and *TGS-Lite+* also require discretised data for short-listing and BIC score calculations. Hence, we apply the same algorithm (namely *2L.wt*) which is used with *TGS* and *TGS+*.

7.6.1 Comparative Study Against a Random Classifier

First, we conduct a preliminary study to verify whether *TGS-Lite* and *TGS-Lite+* perform better than a random classifier. Here, a random classifier represents a classification algorithm that randomly decides whether an edge should be present or absent.

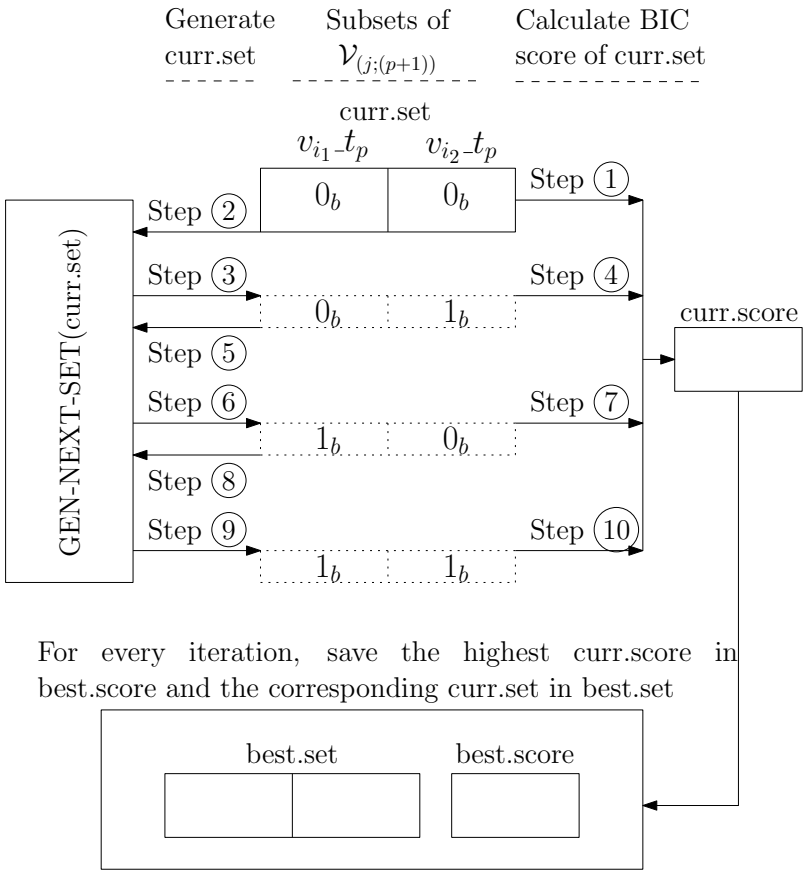


Figure 7.3: Illustration of the Idea for Finding the Highest Scoring Subset amongst $2^{|\mathcal{V}_{(j;(p+1))}|}$ subsets with Only Two Subset-variables, Two Scores and One Subset-generation Script. Each solid-lined rectangle represents a location in the memory. It is assumed that $\mathcal{V}_{(j;(p+1))} = \{v_{i_1}\_t_p, v_{i_2}\_t_p\}$. Therefore, its subsets are $\{\emptyset, \{v_{i_2}\_t_p\}, \{v_{i_1}\_t_p\}, \{v_{i_1}\_t_p, v_{i_2}\_t_p\}\}$. Each subset is represented as a binary string of Boolean TRUE (1_b) and FALSE (0_b) values, indicating whether a gene is present or not in that subset, respectively. Initially, the empty subset is stored in location 'curr.set' (current subset). Please note that curr.set holds the subset itself and not a pointer to the subset. In step 1, the score of the empty subset is calculated and stored inside 'curr.score' (current subset's score which is a floating point number). Subsequently, the values of curr.set and curr.score are copied into 'best.set' (hitherto best subset) and 'best.score' (hitherto best score), respectively. In step 2, the value of curr.set is sent to the subset-generation script, namely 'GEN-NEXT-SET'. In step 3, the script generates the next value of curr.set i.e. the second subset (indicated by the dotted-lined rectangle). In step 4, the score of the second subset is calculated and stored inside curr.score. If curr.score is greater than best.score, then the values of best.score and best.set are replaced with that of curr.score and curr.set, respectively. Otherwise, if curr.score is less than or equal to best.score, then best.score and best.set are kept unchanged. In step 5, the value of curr.set is sent to the script, which generates the next value of curr.set in step 6. The same procedure is followed till all the subsets are exhausted. As a result, the final value of best.set represents the highest scoring subset (ties between two equally scoring subsets are broken in favour of the subset with lower index). Overall, this strategy requires two subset-variables (curr.set and best.set), two scores (curr.score and best.score) and one subset-generation script (GEN-NEXT-SET) to be held in memory. It also requires corresponding gene expression data (not shown in figure) to be stored in memory for calculating the scores.

Predictions of a random classifier tend to result in a line represented by equation 'True Positive Rate = False Positive Rate' (TPR = FPR); here, TPR = Recall; FPR = (FP / (FP + TN)) (Liu et al., 2016). From Figure 7.4 , it is observed that the plots of *TGS-Lite* and *TGS-Lite+* always stay above the random classifier's line. It signifies that both *TGS-Lite* and *TGS-Lite+* perform better than a random classifier on the benchmark datasets.

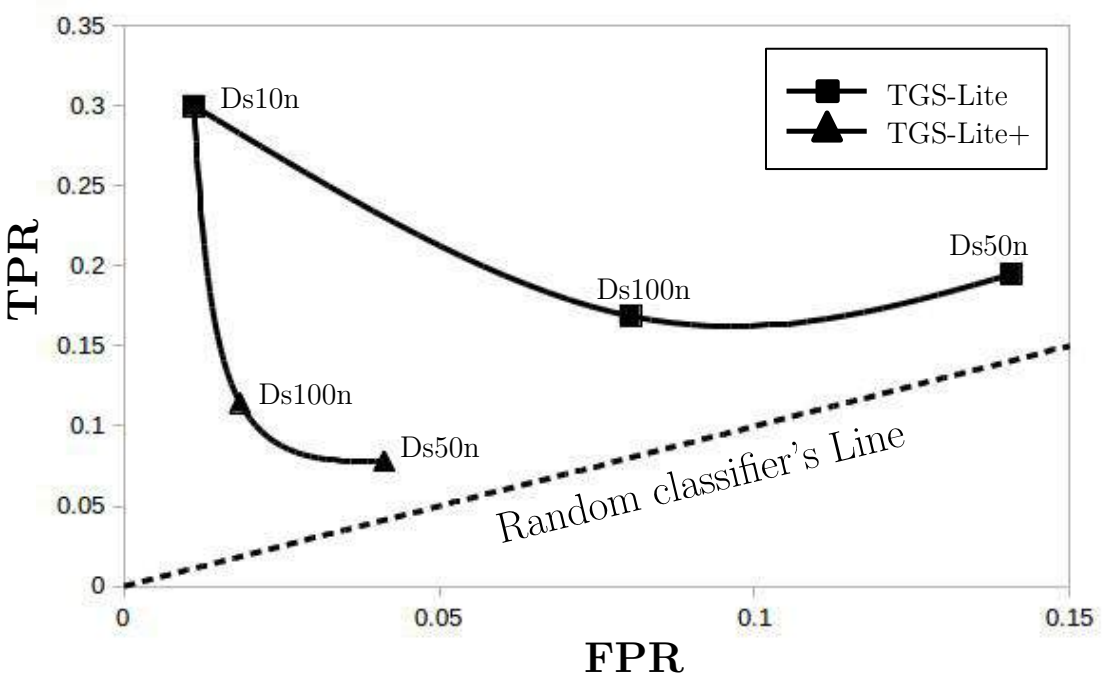


Figure 7.4: The TPR-vs-FPR Plots of *TGS-Lite*, *TGS-Lite+* and a random classifier. Here, TPR = True Positive Rate, FPR = False Positive Rate. The results of *TGS-Lite* for three distinct benchmark datasets are represented as three black squares. These squares are connected (interpolated) with a smooth line (Software used: LibreOffice Calc Version 5.1.6.2; Line type = Cubic spline, Resolution = 20; OS: Ubuntu 16.04.5 LTS). On the other hand, the results of *TGS-Lite+* for three distinct benchmark datasets are represented as three black triangles (the triangle for Ds10n overlaps with the square for Ds10n and hence not visible). These triangles are also connected with a smooth line.

7.6.2 Comparative Study Against Alternative Algorithms

Following the success against a random classifier in the preliminary study, we conduct a comparative study against the alternative algorithms, namely *TGS, TGS+, ARTIVA, TVDBN-0, TVDBN-bino-hard* and *TVDBN-bino-soft*.

Naming Conventions: For *TGS, TGS+, TGS-Lite* and *TGS-Lite+*, some extensions are used with their names to imply different settings of these algorithms. If no extensions are used, then it implies a serial execution of that algorithm with M_f set to 14. On the other hand, if an extension of 'mf[X]' is used, then it implies that M_f is set to 'X'. For example, *TGS.mf24* implies a serial execution of *TGS* with M_f set to 24. Lastly, an extension of 'p[X]' implies a multicore-parallelised execution with the number of computing cores set to 'X'. An example would be *TGS-Lite.p10*, which implies a parallel execution of TGS-Lite with 10 cores.

Lighter Memory Footprint: To compare memory footprints of *TGS* and *TGS-Lite*, an experiment is performed with dataset Ds100n for $M_f = 24$. From Figure 7.5 ,

it is observed that *TGS* occupies 54.7% (column '%MEM') of the memory ($\sim$ 32 GB) within first 1 min 35.25 seconds (column 'TIME+') of execution. Its memory requirement rises further as the time goes on until it reaches a 'thrashing' state (Silberschatz et al., 1991). At that point, we terminate the process. On the other hand, *TGS-Lite* occupies only 0.7% of the memory during the same time point. Moreover, it completes execution without any issues. This experiment demonstrates the advantage of *TGS-Lite*'s memory-efficiency over that of *TGS*. Additionally, it opens up the possibility of parallel execution with *TGS-Lite*. Since the serial execution of *TGS-Lite* only requires 0.7% of the memory, we can execute it in parallel over at most $\lfloor(100/0.7)\rfloor \simeq 142$ cores who share the same memory. The parallelised execution is expected to reduce the runtime by at most a factor of 142. *TGS*, on the other hand, can not take advantage of multicore-parallelisation schemes, since its serial execution alone has such a heavy memory footprint.

Percentage of Memory Usage with dataset Ds100n

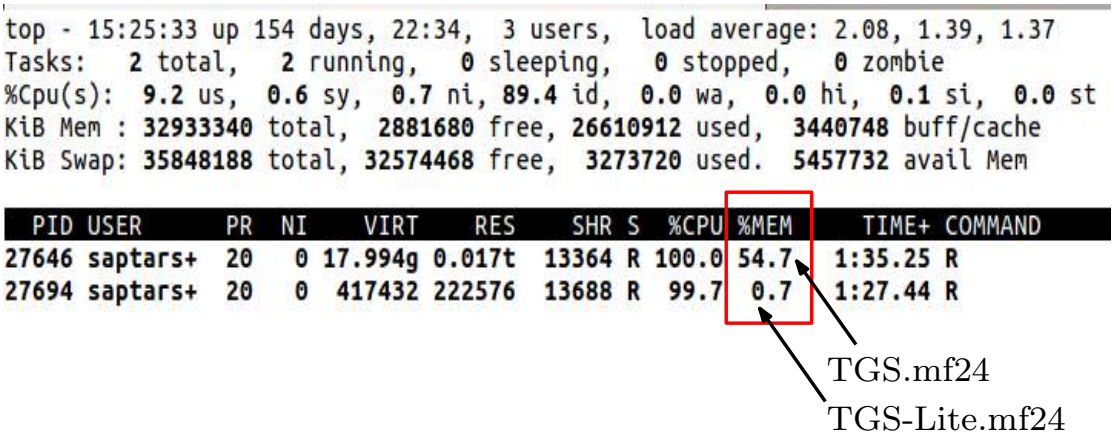


Figure 7.5: Comparative Memory Requirements of the *TGS* and *TGS-Lite* Algorithms. The percentage of memory usage ('%MEM') by *TGS* and *TGS-Lite* when the max fan-in parameter is set to 24. The shown figure is a screenshot of the 'top' command in the Ubuntu OS.

No Loss in Correctness: The memory-efficiency of *TGS-Lite* does not come at the cost of its correctness. It provides the same recall, precision and F1-scores as that of *TGS* (Figure 7.6) [2]. *TGS-Lite+* also meets its expectation by achieving the same recall, precision and F1-scores as that of *TGS+* (Figure 7.6).

Multicore Parallelisation: Although, *TGS-Lite* shares the same time complexity as that of *TGS*, it takes longer runtime than that of *TGS* (Table 7.1). Same is true for *TGS-Lite+*, which takes longer runtime than that of *TGS+* even though they have the same time complexity. The potential cause lies in the difference between their implementations. The strategy which replaces the Bene step in *TGS-Lite* and *TGS-Lite+* is implemented in R programming language, except the function for BIC score calculations which is implemented in C programming language. On the other hand, the Bene step in *TGS* and *TGS+* is implemented in C, which is expected to be significantly faster than the R implementation of the replacement strategy. Hence, a C implementation of the replacement strategy is planned for future.

[2]Please note that algorithms *TVDBN-exp-hard* and *TVDBN-exp-soft* result in error for all datasets.

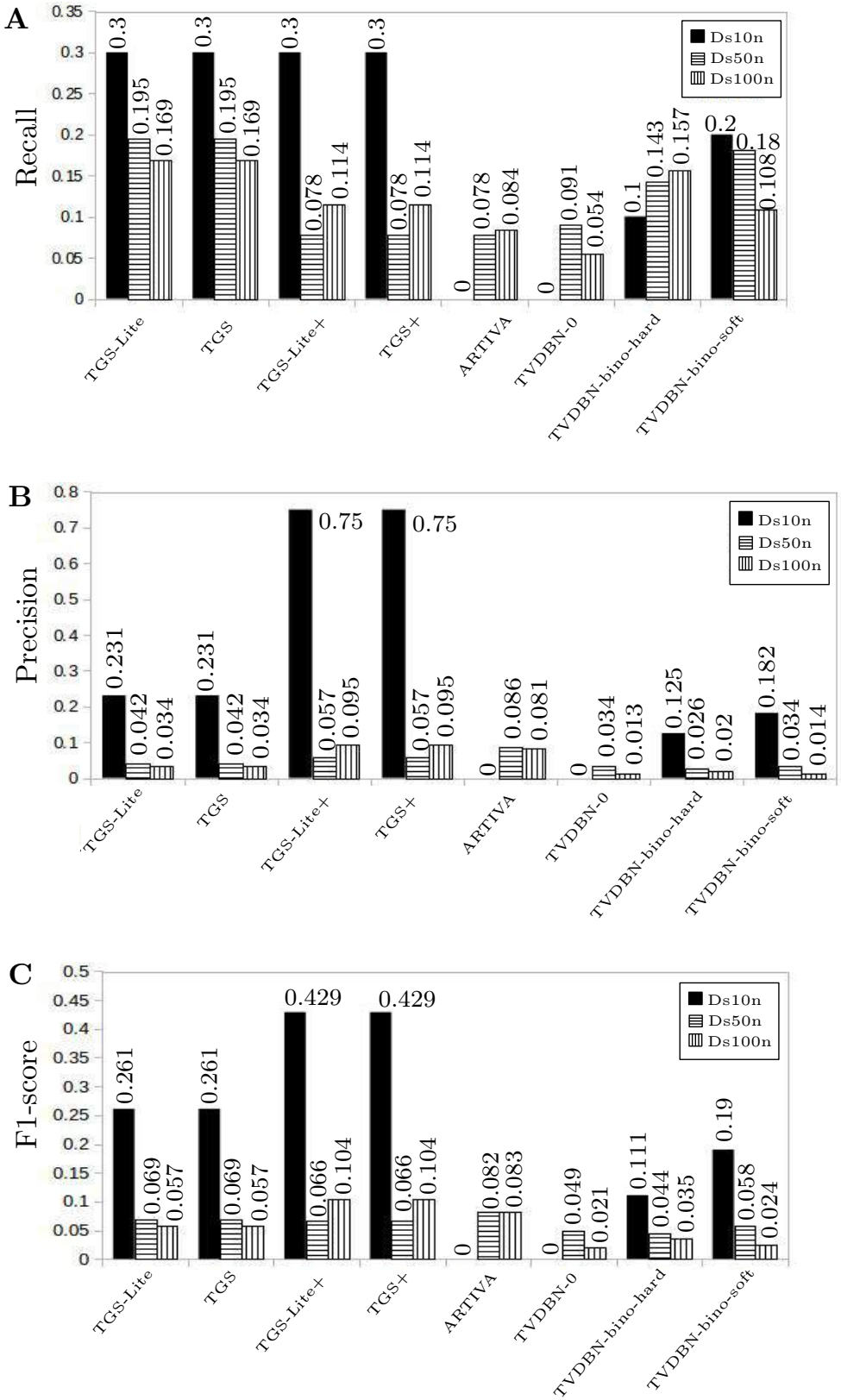


Figure 7.6: Comparative Performance of the Selected Algorithms on the Benchmark Datasets. **A)** Recall, **B)** precision and **C)** F1-scores of the selected algorithms are shown.

At this time, we take advantage of the memory-efficiency of *TGS-Lite* and *TGS-Lite+*. We do that by executing them in parallel over 10 cores. As expected, the runtime reduce drastically compared to that of the serial executions (Table 7.1) [3] .

Table 7.1: Runtime of the Selected Algorithms on the Benchmark Datasets.

Algorithm	**Ds10n**	**Ds50n**	**Ds100n**
TGS-Lite	8.9s	3h 58m 2s	8h 41m 17s
TGS-Lite.p10	6.673s	24m 45s	1h 2m 3s
TGS	5.789s	7m 36s	17m 49s
TGS-Lite+	2.986s	17.383s	12m 12s
TGS-Lite+.p10	6.028s	9.025s	1m 25s
TGS+	5.515s	22.034s	1m 4s
ARTIVA	10m 20s	4h 30m 15s	31h 52m 54s
TVDBN-0	2m 24s	11m 59s	52m 17s
TVDBN-bino-hard	2m 15.2s	9m 38s	2h 53m 32s
TVDBN-bino-soft	2m 14.6s	8m 8s	17m 20s

Effects of Multicore Parallelisation: In this paragraph, we study the effects of multicore parallelisation on the runtime of *TGS-Lite* and *TGS-Lite+*. We use Ds100n for this study as this is the largest benchmark dataset. From Figure 7.7 , it is observed that the runtime reduces strictly as the number of cores increases. The single core represents the serial execution. The 3-core executions are presented to offer a reference point to the users with quad-core processors, where one core can be left out for monitoring purposes. Similarly, the 7-core executions are presented to offer a reference point to the users with octa-core processors.

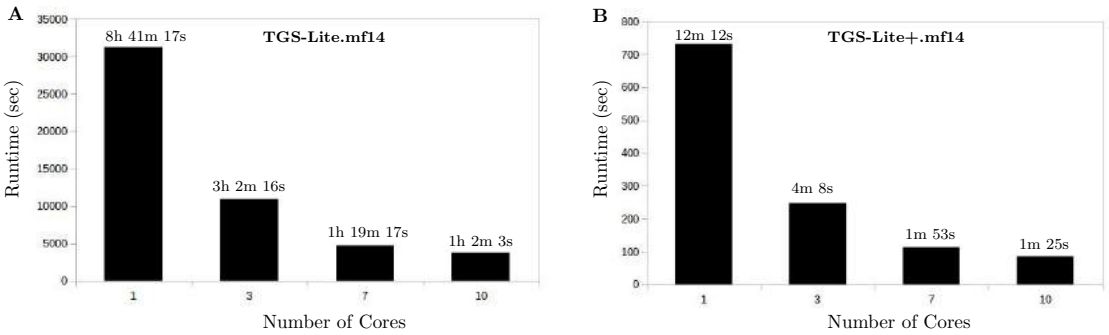


Figure 7.7: The Effects of the Multicore Parallelisation on the Runtime of *TGS-Lite* and *TGS-Lite+*. Dataset Ds100n is used and the max fan-in parameter is set to 14.

Effects of Max Fan-in: Finally, we conclude this section by studying the effects of the max fan-in parameter on the correctness and runtime of the proposed algorithms. The maximum value of the concerned parameter with which *TGS* and *TGS+* are successfully demonstrated is 14. Since *TGS-Lite* and *TGS-Lite+* are significantly more memory-efficient than the aforementioned algorithms, they should be able to run successfully with max fan-in values higher than 14. To verify that, we need to vary the

[3]Please note that algorithms *TVDBN-exp-hard* and *TVDBN-exp-soft* result in error for all datasets.

parameter value from 14 to higher values. These variation can have an effect on the prediction only when at least one gene has more than 14 candidate regulators and the max fan-in restriction is not in place. We find that only dataset Ds100n fulfils this criterion. For this dataset, at least one gene has more than 14 candidate regulators in case of both *TGS-Lite* and *TGS-Lite+* (Table 7.2). Therefore, we study the effects on these algorithms with Ds100n when the parameter value is varied from 14 to 18 (eighteen being the minimum of the maximum numbers of candidate regulators for *TGS-Lite* and *TGS-Lite+*). It is observed that the recall of *TGS-Lite* increases monotonically as the max fan-in increases; at the same time, its precision decreases monotonically; as a result, an upheaval is caused in the F1-scores (Figure 7.8 A). On the other hand, *TGS-Lite+* demonstrates a robust performance as its recall, precision and F1-score remain unchanged for the given range of max fan-in values (Figure 7.8 B). The runtime of both the algorithms increase proportionally to the max fan-in values (Figures 7.8 C and 7.8 D). It is as expected since the higher the max fan-in value, the higher the maximum number of candidate regulators for a gene; therefore, the higher the runtime required for selecting the final set of regulators.

Table 7.2: Maximum Number of Candidate Regulators of a Gene in the *TGS-Lite* and *TGS-Lite+* Algorithms for a given Dataset. These statistics are recorded when the max fan-in restriction is not applied. If the max fan-in restriction is applied, the numbers will be upper bounded by the value assigned to the max fan-in parameter.

Dataset	**TGS-Lite**	**TGS-LIte+**
Ds10n	7	4
Ds50n	33	8
Ds100n	84	18

7.6.3 Comparative Study Against Time-invariant Algorithms

The success against the alternative algorithms encourages us to conduct further comparisons. In this section, we conduct a comparative study against three widely-used algorithms that are not direct alternatives.

The chosen algorithms are not direct alternatives because they can not reconstruct time-varying GRNs. However, they can reconstruct a single time-invariant GRN or 'the summary GRN'. As the name suggests, the summary GRN summarises all regulatory activities by capturing all regulatory edges. On the other hand, it does not capture the temporal order of those edges. For the aforementioned reason, the output of the time-invariant algorithms are not directly comparable with that of the proposed algorithms.

To make them comparable, we utilise the rolled GRN of each proposed algorithm. Consequently, comparisons are performed between the rolled GRN and the summary GRNs of the time-invariant algorithms.

The time-invariant algorithms chosen for this study are: 'B team's algorithm' (hereafter, *BTA*), *GENIE3* and the 'local Bayesian network' (*LBN*). Each of them is chosen for a distinct reason (Table 7.3).

Table 7.3: The Time-invariant Algorithms Selected for the Comparative Study in Section 7.6.3 .

Algorithm	**Reason for selection**
BTA	Relevance. The winning algorithm of the challenge where the benchmark datasets were published.
GENIE3	Popularity. Perhaps, the most widely-used time-invariant algorithm.
LBN	Contemporaneity. Perhaps, the most recently published time-invariant algorithm.

BTA is chosen because of its relevance to the proposed algorithms. The former is the winning algorithm of the DREAM3 In Silico Network Challenge where the benchmark datasets were published (Yip et al., 2010). Therefore, it is interesting to study how the proposed algorithms fare against *BTA* on these datasets.

On the other hand, *GENIE3* is chosen for its popularity. To the best of our knowledge, *GENIE3* is the most widely-used time-invariant algorithm. Its publications combinedly possess 1007 citations in Google Scholar as of 23rd April, 2020 (Huynh-Thu et al., 2010; Aibar et al., 2017).

Finally, *LBN* is chosen for its contemporaneity to the proposed algorithms. The former is perhaps the most recently published time-invariant algorithm (Liu et al., 2016).

The findings of the comparative study against the aforementioned time-invariant algorithms are discussed in the following sub-sections.

7.6.3.1 Comparison with BTA

The first time-invariant algorithm in comparison is *BTA* (Yip et al., 2010). It produces a ranked list of all possible edges. For a dataset with V genes, the list contains $(V\,(V-1))$ edges, when self-loops are not allowed. The higher the rank, the more the confidence of *BTA* on the existence of that edge. Subsequently, a user-defined parameter 'k' is used to select the top k edges and reconstruct a summary GRN. Thus, *BTA* generates $(V\,(V-1))$ summary GRNs by varying the value of k from 1 to $(V\,(V-1))$. These summary GRNs are then compared with the true GRN, if known, to compute an AUROC score (Yip et al., 2010).

On the other hand, each of the proposed algorithms reconstructs a distinct rolled GRN. Therefore, it is infeasible to find out AUROC scores for the proposed algorithms.

To make the proposed algorithms comparable with *BTA*, we design a novel strategy. This strategy requires an in-depth look into the way *BTA* produces the ranked list. The list is generated through seven steps, called 'batch 1' to 'batch 7'. During batches 1 to 6, *BTA* creates a ranked sub-list of potentially true edges. The remaining edges are then appended to the sub-list in batch 7. The experimental results indicate that the initial sub-list contains the strongest predictions whereas batch 7 contains the weakest ones (Yip et al., 2010). For instance, let us consider *BTA*'s predictions with dataset Ds10n, which are referred to as the 'size 10 network' of 'Yeast1' in Yip et al. In batches 1 to 6, *BTA* predicts 32 edges out of which 9 are correct. On the contrary, in batch 7, a total of 58 edges are predicted among which only 1 is correct. Hence, we consider the predictions of *BTA* till batch 6 to generate the summary GRN. It ensures that the weakness of *BTA* is minimized.

With the aforementioned strategy, we extract the number of true positive (TP) and

false positive (FP) predictions for *BTA* from Yip et al. Subsequently, other metrics are calculated from TP and FP with the following formulae.

- **Recall** = (TP / Number of edges in the true GRN).
  Number of edges in the true GRN are 10, 77 and 166 for datasets Ds10n, Ds50n and Ds100n, respectively (Table 4.1).

- **Precision** = 0, if TP = FP = 0.
  Otherwise, precision = (TP / (TP + FP)).

- **F1-score** = 0, if recall = precision = 0 (GERBIL).
  Otherwise, F1-score = the harmonic mean of recall and precision.

The correctness metrics of *BTA* and the proposed algorithms on Ds10n are presented in Table 7.4 . The metrics show that *BTA* achieves the highest number of true positives at the cost of highest number of false positives. At the same time, it provides the highest F1-score, jointly with *TGS+* and *TGS-Lite+*.

Table 7.4: Reconstruction Correctness of the Selected Algorithms on Dataset Ds10n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced.

Algorithm	**TP**	**FP**	**Recall**	**Precision**	**F1**
TGS	3	10	0.3	0.231	0.261
TGS+	3	**1**	0.3	**0.75**	**0.429**
TGS-Lite	3	10	0.3	0.231	0.261
TGS-Lite+	3	**1**	0.3	**0.75**	**0.429**
BTA	**9**	23	**0.9**	0.281	**0.429**

The same pattern is observed for the metrics on Ds50n and Ds100n, presented in Tables 7.5 and 7.6 , respectively.

Table 7.5: Reconstruction Correctness of the Selected Algorithms on Dataset Ds50n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced.

Algorithm	**TP**	**FP**	**Recall**	**Precision**	**F1**
TGS	15	342	0.195	0.042	0.069
TGS+	6	**100**	0.078	0.057	0.066
TGS-Lite	15	342	0.195	0.042	0.069
TGS-Lite+	6	**100**	0.078	0.057	0.066
BTA	**69**	586	**0.896**	**0.105**	**0.189**

Table 7.6: Reconstruction Correctness of the Selected Algorithms on Dataset Ds100n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced.

Algorithm	**TP**	**FP**	**Recall**	**Precision**	**F1**
TGS	28	790	0.169	0.034	0.057
TGS+	19	**181**	0.114	**0.095**	0.104
TGS-Lite	28	790	0.169	0.034	0.057
TGS-Lite+	19	**181**	0.114	**0.095**	0.104
BTA	**145**	2187	**0.873**	0.062	**0.116**

For all the datasets, *BTA* obtains strictly highest numbers of true positives while conceding strictly highest numbers of false positives. However, it monotonically maintains the best balance between the true and false positives, as demonstrated by its F1-scores. Nevertheless, *BTA*'s high F1-scores need further discussions on two aspects.

The first aspect is that the F1-scores stem from summary GRNs. Therefore, no information is provided about the temporal order of the predicted edges. On the contrary, each rolled GRN of the proposed algorithms possesses a distinct sequence of time-varying GRNs. The temporal order of the edges in a rolled GRN can be directly obtained from the corresponding time-varying GRNs.

The second aspect is the data usage. *BTA* utilises two additional data files provided in the DREAM3 challenge. The first file contains 'knock-out' data. These data are produced by knocking out one gene at a time i.e. reducing one gene's expression to zero, and recording its effects on other genes. The second file contains 'knock-down' data. These data are acquired by knocking down one gene at a time i.e. reducing one gene's expression to half of its normal or 'wild-type' expression, and recording its effects on other genes. Usage of the knock-out data significantly enhances *BTA*'s ability for capturing the edges where the source node is the sole regulator of the target node (Yip et al., 2010). Hence, it is unjustified to compare the F1-scores of *BTA* with that of the proposed algorithms, without keeping *BTA*'s additional data usage in mind. At the same time, we recognize *BTA*'s ability to use that data to improve its correctness. Adding such capability to the proposed algorithms can be a worthwhile extension in future.

Finally, we compare the runtime of *BTA* with that of the proposed algorithms (Table 7.7). The runtime of *BTA* is again extracted from Yip et al. *BTA*'s runtime are recorded on a 'high-end cluster' (the cluster configuration is not found). The implementation is written in Java (Gerstein Lab, 2010). On the other hand, the proposed algorithms are implemented in R and their runtime are recorded on 3 GHz CPUs with 32 GB main memory. Therefore, an exact comparison of *BTA*'s runtime with that of the proposed algorithms is not conducted. Instead, the orders of their runtime are compared. It is observed that *BTA*'s runtime are in hours whereas that of the proposed algorithms are in minutes. For example, *BTA* requires 78 hours for Ds100n, while *TGS-Lite+* takes only 12 minutes. Therefore, *BTA* may require months to process datasets with thousands of genes. As a result, usage of *BTA* can be prohibitive for large-scale datasets. Yip et al. points this out to be a major limitation of *BTA*. On the other hand, the proposed algorithms can process such datasets in minutes.

Summary: The findings of the comparison between *BTA* and the proposed algorithms can be summarised as follows:

Table 7.7: Runtime of the Selected Algorithms on the Benchmark Datasets.

Algorithm	**Ds10n**	**Ds50n**	**Ds100n**
TGS-Lite	8.9s	3h 58m 2s	8h 41m 17s
TGS-Lite.p10	6.673s	24m 45s	1h 2m 3s
TGS	5.789s	7m 36s	17m 49s
TGS-Lite+	2.986s	17.383s	12m 12s
TGS-Lite+.p10	6.028s	9.025s	1m 25s
TGS+	5.515s	22.034s	1m 4s
BTA	2m	13h	78h

- *BTA* can achieve monotonically higher F1-scores with the help of additional information, such as - knock-out data.

- *BTA*'s runtime can be prohibitive for its usage with large-scale datasets. The proposed algorithms can process such datasets in convenient time frames.

7.6.3.2 Comparison with GENIE3

The second time-invariant algorithm in comparison is *GENIE3* (Huynh-Thu et al., 2010; Aibar et al., 2017). For this comparison, we use the implementation of *GENIE3* in R package 'GENIE3' (GENIE3). It is executed on the same hardware as that of the proposed algorithms. The steps for reproducing GENIE3's results can be found at: `https://github.com/sap01/Test_GENIE3_supplem` .

Like *BTA*, *GENIE* returns a ranked list of all possible edges. To generate a summary GRN from the ranked list, we devise a novel strategy. The strategy is to select the top 'k' number of edges from the list, where k is the number of edges in the rolled GRN of a proposed algorithm. Then, there would be same number of edges in *GENIE3*'s summary GRN and the rolled GRN. Comparing these GRNs with respect to the true GRN, we can have a fair comparison between their correctness metrics.

Following the aforementioned strategy, we compute the correctness metrics of *GENIE3* on dataset Ds10n (Table 7.8). Since, the rolled GRNs of *TGS* and *TGS-Lite* have 13 edges each, we select top 13 edges from *GENIE3*'s ranked list. The resultant summary GRN is referred to as 'GENIE3.top13'. Hereafter, we follow the naming convention of 'GENIE3.top$k$' to represent the top k edges from *GENIE3*'s ranked list. It is observed that *TGS* and *TGS-Lite* outperform *GENIE3.top13* in both true and false positive predictions.

Similarly, we use *GENIE3.top4* to compare with *TGS+* and *TGS-Lite+*. It is observed that the latter algorithms outperform the former in both true and false positive predictions.

Table 7.8: Reconstruction Correctness of the Selected Algorithms on Dataset Ds10n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced.

Algorithm	**TP**	**FP**	**Recall**	**Precision**	**F1**
TGS	**3**	10	**0.3**	0.231	0.261
TGS-Lite	**3**	10	**0.3**	0.231	0.261
GENIE3.top13	1	12	0.1	0.077	0.087
TGS+	**3**	**1**	**0.3**	**0.75**	**0.429**
TGS-Lite+	**3**	**1**	**0.3**	**0.75**	**0.429**
GENIE3.top4	0	4	0	0	0

For datasets Ds50n and Ds100n, the same pattern is observed; please see Tables 7.9 and 7.10 , respectively. Every proposed algorithm outperforms the corresponding *GENIE3* variant in both true and false positive predictions.

Table 7.9: Reconstruction Correctness of the Selected Algorithms on Dataset Ds50n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced.

Algorithm	**TP**	**FP**	**Recall**	**Precision**	**F1**
TGS	**15**	342	**0.195**	0.042	**0.069**
TGS-Lite	**15**	342	**0.195**	0.042	**0.069**
GENIE3.top357	9	348	0.117	0.025	0.041
TGS+	6	**100**	0.078	**0.057**	0.066
TGS-Lite+	6	**100**	0.078	**0.057**	0.066
GENIE3.top106	2	104	0.026	0.019	0.022

Table 7.10: Reconstruction Correctness of the Selected Algorithms on Dataset Ds100n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced.

Algorithm	**TP**	**FP**	**Recall**	**Precision**	**F1**
TGS	**28**	790	**0.169**	0.034	0.057
TGS-Lite	**28**	790	**0.169**	0.034	0.057
GENIE3.top818	20	798	0.12	0.024	0.041
TGS+	19	**181**	0.114	**0.095**	**0.104**
TGS-Lite+	19	**181**	0.114	**0.095**	**0.104**
GENIE3.top200	8	192	0.048	0.04	0.044

Finally, we investigate the runtime of *GENIE3* (Table 7.11). It is found that *GENIE3* is significantly faster than the *TGS-Lite* variants. However, other proposed algorithms are observed to be as fast as *GENIE3*.

Table 7.11: Runtime of the Selected Algorithms on the Benchmark Datasets.

Algorithm	**Ds10n**	**Ds50n**	**Ds100n**
TGS-Lite	8.9s	3h 58m 2s	8h 41m 17s
TGS-Lite.p10	6.673s	24m 45s	1h 2m 3s
TGS	5.789s	7m 36s	17m 49s
TGS-Lite+	2.986s	17.383s	12m 12s
TGS-Lite+.p10	6.028s	9.025s	1m 25s
TGS+	5.515s	22.034s	1m 4s
GENIE3	0.906s	1m 51s	14m 5s

Summary: The findings of the comparison between *GENIE3* and the proposed algorithms can be summarised as follows:

- *GENIE3* is significantly faster than *TGS-Lite*. However, other proposed algorithms are as fast as *GENIE3*.
- Every proposed algorithm outperforms *GENIE3* in true and false positive predictions on all benchmark datasets.

7.6.3.3 Comparison with LBN

The third and final time-invariant algorithm in comparison is *LBN* (Liu et al., 2016). Unfortunately, *LBN*'s authors informed us that its source code is not available. Hence, experimental comparison between *LBN* and the proposed algorithms can not be performed. However, *LBN* is methodologically the most similar time-invariant algorithm to the proposed time-varying ones. Therefore, it is interesting to comparatively study their methodologies.

LBN follows a two-step framework. In the first step, a shortlist of candidate regulators is prepared for a given regulatee gene. Consequently in the second step, the shortlisted candidates are further analysed to prepare the final list of regulators. The proposed algorithms follow the same two-step framework.

In *LBN*, the shortlisting step is based on mutual information. Only those genes are shortlisted whose mutual information with the regulatee gene are greater than or equal to a user-defined threshold α. In the proposed algorithms, the shortlisting step is based on mutual information. However, it is not dependent on any user-defined threshold. Instead, only those genes are selected whose mutual information are statistically significant with the regulatee gene. The statistical significance is tested with a well-established method named *CLR* (Faith et al., 2007). This method is data-driven and does not require any user input. On the other hand, it is demonstrated that the correctness of *LBN* varies significantly when the value of α is varied (Liu et al., 2016). Hence, it is difficult for the users to select an appropriate value of α in case of real datasets with no such information known a priori. The proposed algorithms overcome this limitation by automating the shortlisting decision.

The final-selection step in the proposed algorithms is based on local Bayesian networks. In this step, they construct a local Bayesian network for each regulatee gene. The local Bayesian network is comprised of the regulatee gene and its finalised regulators. Then all such local Bayesian networks are united to construct the final network.

LBN follows the same procedure. However, the united network is not the final network in *LBN*. Instead, *LBN* conducts further refinements on the united network. These refinements are again dependent on a user-defined parameter β.

Moreover, *LBN* performs the complete two-step process iteratively until the output network of the current iteration is unchanged from that of the last iteration. As a result, the time complexity of *LBN* is dependent on the number of iterations. If the number of iterations is l, then *LBN*'s time complexity is $\mathcal{O}\big(2 \times l \times V^2 + l \times V + l \times V \times 2^m\big)$, where V = the total number of genes and m = the number of regulator genes (Liu et al., 2016). When any gene can be a regulator gene i.e. $m = V$, the time complexity transforms to $\mathcal{O}\big(2 \times l \times V^2 + l \times V + l \times V \times 2^V\big)$.

In case, V is significantly larger than l, *LBN*'s time complexity reduces to $\mathcal{O}\big(V \times 2^V\big)$. On the other hand, the time complexities of *TGS* and *TGS-Lite* are $o\big(V^2 \times \lg V\big)$; that of *TGS+* and *TGS-Lite+* are $\mathcal{O}\big(V^3\big)$ (Pyne and Anand, 2019a).

Summary: The findings of the comparison between *LBN* and the proposed algorithms can be summarised as follows:

- The correctness of *LBN* depends upon two user-defined parameters, namely α and β. Their values are used as thresholds for the significance test of genes' mutual information. In case of real datasets with no such information known a priori, it is difficult for the users to set appropriate values of these parameters. On the contrary, the proposed algorithms employ a well-established method named *CLR*. This method is data-driven and does not require any user input. Thus, the proposed algorithms overcome user-dependency for the significance test of genes' mutual information.
- *LBN* is an iterative-convergent algorithm. The proposed algorithms are non-iterative.
- *LBN* has a higher time complexity than that of the proposed algorithms. For a dataset with V number of genes, the time complexity of *LBN* is $\mathcal{O}\big(V \times 2^V\big)$, the time complexities of *TGS* and *TGS-Lite* are $o\big(V^2 \times \lg V\big)$, and that of *TGS+* and *TGS-Lite+* are $\mathcal{O}\big(V^3\big)$.

7.6.4 Results with a Large-scale Dataset

Following favourable results with the benchmark datasets, we apply the proposed algorithms on a large-scale dataset. This dataset contains time-series expressions of 4028 genes acquired throughout the life cycle of fruit flies (Drosophila melanogaster, in short, Dm) (Arbeitman et al., 2002). However, the true time-varying GRNs are unknown. Moreover, the summary GRN is also unknown. As a consequence, it is not possible to compute correctness of the reconstructed networks. Hence, we follow the strategy devised by Song et al. (Song et al., 2009a). They select a subset of 588 genes. These genes are known to have regulatory activities during the development of fruit flies. Consequently, Song et al. investigate whether the regulatory activities of these genes in the reconstructed networks match with their known activities.

Using the aforementioned strategy, we extract a smaller dataset named 'Drosophila melanogaster life cycle 3' or DmLc3 (Pyne et al., 2020). It consists of four sub-datasets corresponding to four stages of fruit fly's life cycle. The sub-datasets are: DmLc3E (embryo), DmLc3L (larva), DmLc3P (pupa) and DmLc3A (adult). They contain 6, 2, 3 and 2 time points, respectively (Table 7.12).

The proposed algorithms and their alternative algorithms are applied on the DmLc3 sub-datasets. Among the alternative algorithms, we exclude *TVDBN-exp-hard* and

Table 7.12: A Summary of the DmLc3 Sub-datasets.

Sub-dataset	Number of Genes	Number of Time Points	Number of Time Series
DmLc3E	588	6	5
DmLc3L	588	2	5
DmLc3P	588	3	6
DmLc3A	588	2	4

TVDBN-exp-soft since they fail to process any of the benchmark datasets. A comparative study of the selected algorithms' results is presented in the following sections.

7.6.4.1 Better Memory Management

We observe that *TVDBN-0*, *TVDBN-bino-hard* and *TVDBN-bino-soft* fail to process any of the sub-datasets due to memory-management issues. On the other hand, *ARTIVA* is able to process DmLc3E. However, it fails to process other sub-datasets. Although, the failure is not due to memory-management; it is potentially due to *ARTIVA*'s implementation. On the contrary, the proposed algorithms are able to process all sub-datasets without any issues.

7.6.4.2 Superior Correctness

Evaluation Strategy: For evaluating correctness, we select a subset of 25 genes. These genes are known regulators in the development process (Marbach et al., 2012). For each of these regulators, we ask the following two questions :

- Q. Does the regulator have any regulatees in the expected stages of the reconstructed GRNs?
- Q. If yes, are they known or potentially true regulatees of that regulator?

We evaluate how successfully a GRN is able to answer these questions. Based on that, we measure the correctness of that GRN and compare it with that of the other GRNs. The information about the true regulatees of a regulator is retrieved from a biological knowledge base. This knowledge base is known as 'TRANSFAC public database version 7.0' (GmbH). It is claimed to be a gold standard in the area of gene regulation (Gen).

Selection of Max Fan-in: For *TGS* and *TGS+*, we retain the max fan-in value at 14 to avoid segmentation faults (Chapters 5 and 6). Hence, we refer to them as *TGS.mf14* and *TGS+.mf14*, respectively. On the other hand, *TGS-Lite* and *TGS-Lite+* do not require that restriction. Hence, their max fan-in value is incremented to 15. The objective behind this increment is to find out whether they can capture potentially more true edges than that of *TGS.mf14* and *TGS+.mf14*. Following the naming convention, the former algorithms are referred to as *TGS-Lite.mf15* and *TGS-Lite+.mf15*, respectively.

Evaluation: It is found that *ARTIVA* predicts potentially the least number of false positive edges. The term 'potentially' refers to the fact that biological knowledge about the existence or non-existence of an edge is incomplete and continually growing. Therefore, an edge which is yet not known to exist, may later found to exist. On the other

hand, *ARTIVA* captures potentially the least number of true positive edges. *TGS.mf14* and *TGS-Lite.mf15* capture larger numbers of true positive edges than that of *ARTIVA*. Among themselves, *TGS-Lite.mf15* captures more number of true positives than that of *TGS.mf14* due to the higher max fan-in. Nevertheless, both of them make higher false positive predictions in comparison to *ARTIVA*. In contrast, *TGS+.mf14* and *TGS-Lite+.mf15* are able to find a balance. They incur less numbers of false positives than that of *TGS.mf14* and *TGS-Lite.mf15*. At the same time, they capture larger numbers of true positives than that of *ARTIVA*. For the brevity of discussion, we present some of the findings in the following paragraphs. The complete set of findings are depicted in Pyne et al. (Pyne and Anand, 2019a).

In this paragraph, we present the results related to gene 'Antp'. It is one among the selected 25 regulators. 'Antp' is known to be essential in defining embryonal segment identity. In the embryonic stage, *ARTIVA* predicts 'Antp' to have two regulatees – 'hdc' and 'Sulf1'. For 'hdc', there does not exist sufficient information to validate its relationship with 'Antp'. On the contrary, for 'Sulf1', the existing biological knowledge can be used to argue that it is potentially a false positive prediction. To have a regulatory relationship, 'Sulf1' is expected to be localised in the same cellular regions as that of 'Antp'. However, 'Sulf1' is localised in extracellular regions, whereas 'Antp' is not. Both of the aforementioned regulatees are rejected by *TGS-Lite.mf15*. Instead, it selects 'opa' to be a regulatee of 'Antp'. This is highly likely to be a true positive prediction, given the facts: (a) 'opa' is co-localised with 'Antp' in nucleus, and (b) 'opa' is known to be involved in the development of segmented embryos. *TGS-Lite.mf15* predicts 17 more regulatees in the embryonic stage. *TGS.mf14* predicts a set of 15 regulatees which is a proper subset of that of *TGS-Lite.mf15*. Most of these predictions are false positives. They are avoided by *TGS+.mf14* and *TGS-Lite+.mf15*. On the other hand, *TGS-Lite+.mf15* agrees with the prediction of 'opa'. It enhances the confidence on 'opa' to be a regulatee of 'Antp' during the embryonic stage. Experimental validation of this relationship can be considered in future.

In this paragraph, we discuss the results corresponding to gene 'eve'. It is one among the selected 25 regulators. 'eve' contributes to the development of the central nervous system. *ARTIVA* does not predict any regulatees for 'eve' in the embryonic stage. On the other hand, *TGS.mf14* and *TGS-Lite.mf15* predict three regulatees each. Among them, two regulatee are in common; they are – 'twi' and 'inv'. Both of them are potentially true positive predictions. The support for 'twi' are: (a) it is co-localised with 'eve' in nucleus, and (b) they co-participate in cell organisation/biogenesis. The support behind 'inv' comes from the fact that both 'inv' and 'eve' are essential during the segmentation of embryos. 'inv' is missed by *TGS+.mf14* and *TGS-Lite+.mf15*; however, they successfully capture 'twi', which is also the sole regulatee predicted by them.

In this paragraph, we discuss the results related to gene 'prd'. It is one among the selected 25 regulators. 'prd' plays an important role in the development of embryos. *ARTIVA* predicts that 'prd' has a single regulatee named 'LIMK1' in the embryonic stage. This is potentially a false positive prediction since 'prd' is localised in nucleus and 'LIMK1' is not. 'LIMK1' is rejected by *TGS.mf14* and *TGS-Lite.mf15*. Instead, they select 'eve' which is a known regulatee of 'prd'. At the same time, *TGS.mf14* and *TGS-Lite.mf15* predict seven and eight more regulatees of 'prd', respectively. Most of them are potentially false positive predictions. These predictions are avoided by *TGS+.mf14* and *TGS-Lite+.mf15*. Instead, they predict only a single regulatee named 'Knrl'. This is potentially a true positive prediction since 'Knrl' is co-localised with 'prd' in nucleus.

Thus, *TGS+.mf14* and *TGS-Lite+.mf15* avoid potential false positive regulatees of *TGS.mf14* and *TGS-Lite.mf15*. Simultaneously, they capture potentially more true positive regulatees than that of *ARTIVA*.

7.6.4.3 Higher Speed

Runtime can be deciding factor when large-scale datasets are concerned. In that regard, the proposed algorithms provide significantly lower runtime than that of *ARTIVA* (Table 7.13) [4] . Among the proposed algorithms, *TGS+.mf14* is proved to be the fastest. Inspired by its speed, we apply *TGS+.mf14* on the parent dataset with all 4028 genes (Pyne et al., 2020). *TGS+.mf14* is able to process the data of the embryo stage, the longest stage, in only 44 minutes.

Table 7.13: Runtime of the Selected Algorithms on the DmLc3 Sub-datasets. In each column, the lowest runtime is boldfaced.

Algorithm	**DmLc3E**	**DmLc3L**	**DmLc3P**	**DmLc3A**
TGS.mf14	22m 28s	4m 12s	10m 49s	3m 51s
TGS-Lite.mf15	3h 54m 8s	41m 7s	1h 43m 31s	44m 41s
TGS+.mf14	**22s**	**22s**	**22s**	**21s**
TGS-Lite+.mf15	22m 21s	24m 17s	25m 30s	29m 18s
ARTIVA	6h 59m 28s	err	err	err

From this study, it can be concluded that the proposed algorithms are the preferred choice for processing large-scale datasets in terms of memory-management, correctness and speed.

7.7 Excerpt and Future Work

In this chapter, we present two novel algorithms for reconstructing time-varying gene regulatory networks from time-series gene expression datasets. The first algorithm is named 'an algorithm for reconstructing Time-varying Gene regulatory networks with Shortlisted candidate regulators - which is Light on memory' (*TGS-Lite*). This algorithm jointly provides state-of-the-art recall and time complexity with the *TGS* algorithm, which is proposed in Chapter 5 . However, *TGS-Lite* offers the aforementioned advantages while also requiring significantly less memory than that of *TGS*. For these reasons, the user can consider *TGS-Lite* to be the best choice when the objective is to maximise true positive predictions at an acceptable cost of incurring false positive predictions. One such application can be the discovery of novel biomarkers.

The second algorithm is named 'TGS-Lite Plus' (*TGS-Lite+*). It jointly offers state-of-the-art F1-score with the *TGS+* algorithm, which is proposed in the previous chapter (Chapter 6). Moreover, *TGS-Lite+* does so with a significantly less memory requirement than that of *TGS+*. Although *TGS-Lite+*'s time complexity is slightly higher than that of *TGS-Lite*, the experimental results demonstrate that its runtime is competitive to that of *TGS-Lite*. Therefore, the user can consider *TGS-Lite+* to be the best choice

[4]Please note that algorithms *TVDBN-bino-hard* and *TVDBN-bino-soft* result in error for all sub-datasets.

when the objective is to maximise F1-score i.e. to reconstruct the networks as correctly as possible.

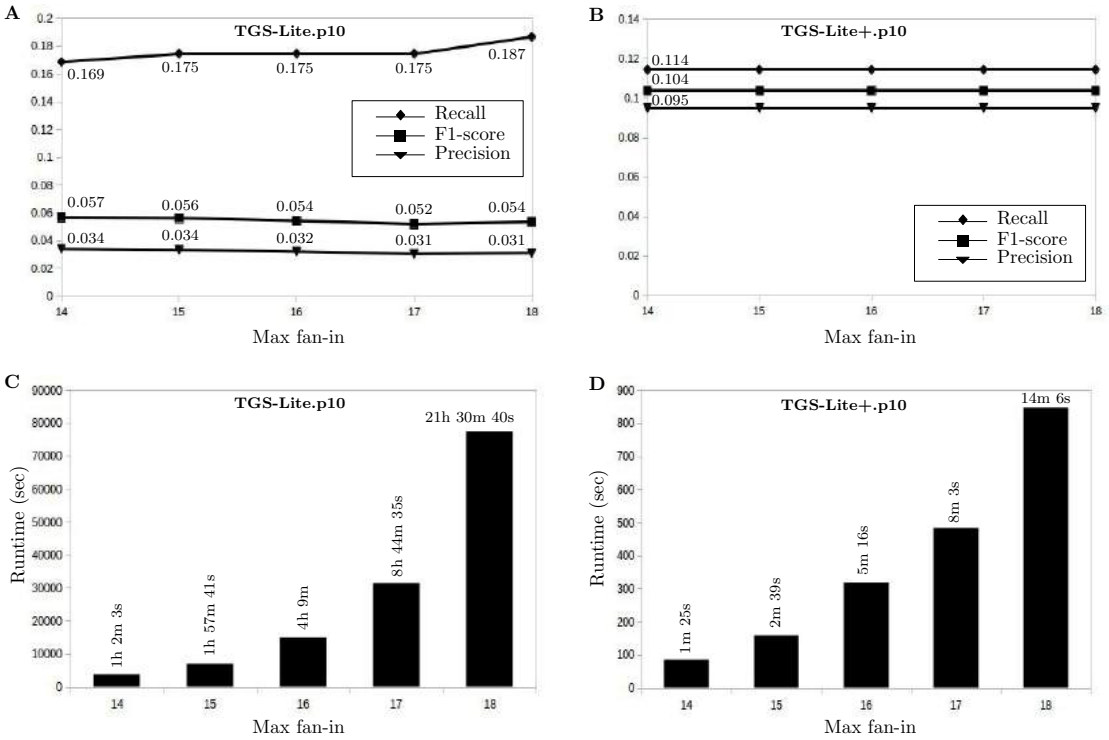


Figure 7.8: The Effects of the Max Fan-in Parameter on the Correctness (**A, B**) and Runtime (**C, D**) of the TGS-Lite and TGS-Lite+ Algorithms. Dataset Ds100n is used and number of cores is set to 10 for multicore parallelisation.

Nevertheless, there are scopes for improvements. The reconstruction problem can be viewed as a collection of smaller sub-problems. Each sub-problem corresponds to predicting the regulators of a distinct gene at a particular time interval.

To solve each sub-problem, both the proposed algorithms follow a two-step pipeline. In the first step, they produce a time-invariant shortlist of candidate regulators for the concerned gene.

In the next step, they select a subset of regulators from the shortlist. This subset is specific to a particular time interval since the selection is based on that time interval's gene expression data.

Therefore, the time-invariant shortlist is used to predict the time-varying final lists. Since the shortlist is time-invariant i.e. based on all time intervals, it tends to favour the candidates who are active for a large number of time intervals.

Thus, this short-listing strategy may overlook the candidates who are active during a small number of time intervals. Such scenarios are common in case of biological networks where a transient regulator can become active for a small duration and cause a large cascading effect.

Therefore, being able to identify transient regulators is crucial for understanding the underlying mechanism. However, if such a regulator is omitted from the shortlist, there are no means to capture that regulator in the final list of the concerned time interval.

For that reason, we aim to develop a short-listing strategy where shortlists are specific to every time interval.

7.8 Contributions

This work is published in IEEE/ACM Transactions on Computational Biology and Bioinformatics (Pyne and Anand, 2019a).

7.9 Chapter Summary

In the previous chapters, we propose two reconstruction algorithms named *TGS* and *TGS+* (Chapters 5 and 6). These algorithms offer time-efficiency compatible with large-scale datasets. Additionally, *TGS+* provides correctness competitive to that of *ARTIVA*. However, both *TGS* and *TGS+* lack in memory-efficiencies required to process large-scale datasets. In this chapter, we extend *TGS* to *TGS-Lite*, and *TGS+* to *TGS-Lite+*. The extended algorithms offer the same time complexity and correctness as that of their original variants. Moreover, the former algorithms provide memory-efficiencies compatible with large-scale datasets. Nonetheless, all four proposed algorithms tend to miss the edges that remain active for a small number of time intervals. Such edges, known as 'transient edges', can be crucial for understanding the regulatory mechanism. Hence, we aim to mitigate this limitation in the next chapter.

Chapter 8

Capturing Transient Edges

In the previous chapters, three contributions are made towards efficient reconstruction of time-varying Gene Regulatory Networks (GRNs) from time-series gene expression data. The first contribution is proposing 'an algorithm for reconstructing Time-varying Gene regulatory networks with Shortlisted candidate regulators', in short, *TGS*; it demonstrates the state-of-the-art time-efficiency (Chapter 5). In the second contribution, we propose the 'TGS Plus' algorithm (*TGS+*); it maintains the time-efficiency of *TGS* while achieving a better balance between recall and precision (Chapter 6). The third contribution introduces the third set of algorithms. The set contains two algorithms – 'TGS - which is Light on memory' (*TGS-Lite*) and 'TGS-Lite Plus' (*TGS-Lite+*). They offer the same time-efficiencies and correctnesses of *TGS* and *TGS+*, respectively, in a significantly more memory-efficient manner (Chapter 7).

These algorithms are also demonstrated to provide state-of-the-art recalls with respect to three benchmark datasets. Nevertheless, the recall values are far from the desired value of '1' (the highest being 0.3). One potential reason behind such low recalls lies in the algorithm design. By design, the proposed algorithms tend to overlook 'transient edges' i.e. the edges that remain active for short periods of time. Although such edges are short-lived themselves, they can trigger long-lasting down-stream effects. Therefore, capturing transient edges is crucial for understanding the regulatory mechanism underlying the data. In this chapter, we design the fourth set of algorithms that combines the efficiency of the previously proposed algorithms with a superior ability to capture transient edges.

The chapter is organised into multiple sections. In Section 8.1 , we discuss the limitations of the previously proposed algorithms. Subsequently, in Section 8.2 , a novel idea is presented for overcoming these limitations. Based on the novel idea, we propose four novel algorithms in Section 8.3 . Sections 8.4 and 8.5 discuss the experimental setup and corresponding results. An excerpt along with a pointer to the future work are provided in Section 8.6 . The research contributions are acknowledged in Section 8.7 . Finally, in Section 7.9 , a summary of the chapter is presented.

8.1 Limitations of the Previously Proposed Algorithms

The objective of the previously proposed algorithms is to identify the regulators of each gene during every time interval. These algorithms accomplish the objective by employing a two-step framework. In the first step, they generate a shortlist of candidate regulators for each gene. These shortlists are time-invariant. In the second step, they select a subset of the shortlisted regulators for each time interval. This subset represents the

final set of regulators of the concerned gene during that particular time interval.

Thus, the final set of regulators of a gene may vary for different time intervals. However, all such final sets are subsets of the same shortlist, which is specific to the gene but invariant to time. The shortlist is time-invariant because the framework uses the whole time-series dataset to compute the mutual-information values of other genes with the concerned gene; subsequently, those genes which share statistically significant mutual information with the concerned gene are shortlisted. On the other hand, when selecting the final set for a particular time interval, the framework utilises the data specific to that time interval.

Therefore, the genes that do not share significantly high mutual information across the whole time series with the concerned gene are less likely to be shortlisted as the candidate regulators of the latter gene. This strategy is useful for rejecting the genes that have no regulatory effects on the concerned gene during any interval. However, the strategy may also reject the genes that have regulatory effects on the concerned gene during a small number of time intervals. Such 'transient' regulators may not get shortlisted. In that case, those transient edges are not captured in the reconstructed GRNs (Figure 8.1).

8.2 A Novel Idea for Overcoming the Limitations

We can capture the transient edges if we consider time interval-specific mutual information. To do that, a simple idea is to generate one shortlist for each time interval with respect to the concerned (regulatee) gene. That shortlist should represent the candidate regulators who share a statistically significant mutual information with the latter gene during that particular time interval (Figure 8.2).

8.3 Design of Novel Algorithms Based on the Novel Idea

Based on the idea of time-varying short-listing, we develop newer algorithms. These algorithms follow the same two-step framework of first short-listing and then finalising. However, the first step is modified to produce time interval-specific shortlists.

We continue using the measure of pairwise mutual information for the task of short-listing. Suppose, we want to shortlist the candidate regulators of gene v_2 during time-interval (t_1, t_2). Assuming that the regulatory activities follow a first-order Markovian process, the task translates to selecting a subset of genes whose expressions at time-point t_1 share statistically significant mutual information with the expression of v_2 at time-point t_2. In the previously proposed algorithms, the statistical significance of a mutual information value is determined with the *CLR* algorithm (Faith et al., 2007). However, *CLR* is not directly compatible with the scenario where the genes in the given pair belong to two different time points. For that purpose, we develop a variant of *CLR*, namely 'CLR - which is Time-varying', in short, the *CLR-T* algorithm. An example of how *CLR-T* works is demonstrated in Algorithm 6 .

Thus, we modify the first step of the existing framework by replacing *CLR* with *CLR-T*. As a result, the modified framework generates time-varying shortlists of candidate regulators in the first step; then in the second step, the finals sets of regulators are produced from the corresponding shortlists (Figure 8.3).

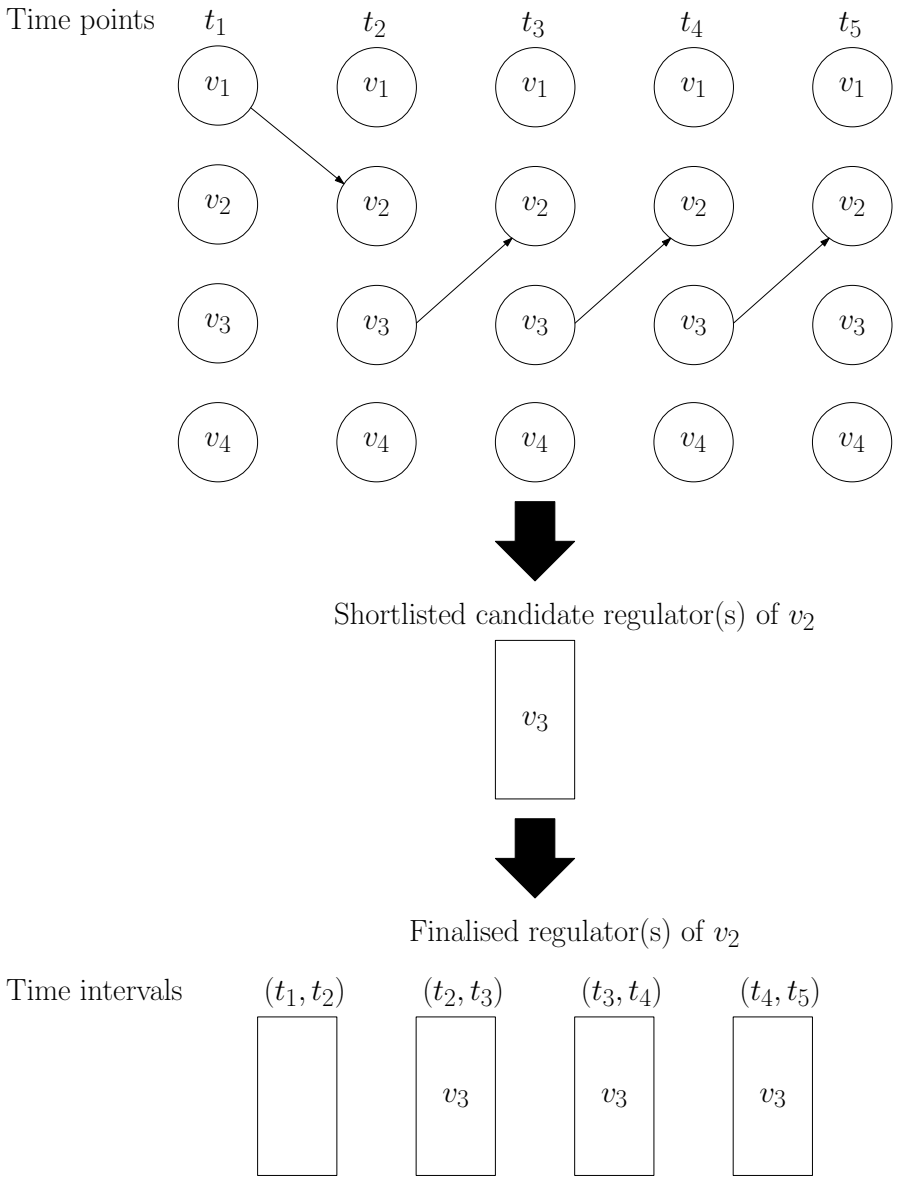


Figure 8.1: An Example for Illustrating the Limitations of the Previously Proposed Algorithms. In this example, a system with four genes – $\{v_1, \cdots, v_4\}$ – is considered. Among them, we arbitrarily choose v_2 whose regulator(s) we want to predict. The true regulator(s) of v_2 during different time intervals are shown with the directed edges, across five time points – $\{t_1, \cdots, t_5\}$. We assume that these relationships are reflected in a time-series dataset. Given that dataset, the previously proposed algorithms generate a shortlist of candidate regulators. Since, v_2 itself and v_4 have no regulatory relationship with v_2 in any of the time intervals, they are expected to share a low mutual information with v_2; thus, they are correctly rejected. On the other hand, v_1 is expected to share a high mutual information with v_2 during the first time interval and low during all other time intervals. Therefore, when mutual information are computed from the whole dataset, v_1 is less likely to be shortlisted. However, v_3 who is expected to share high mutual information with v_2 across most time intervals, is most likely to get shortlisted. If v_1 is not shortlisted, it is definitely not going to appear in the final list. Thus, the previously proposed algorithms are not able to capture the transient edge from gene v_1 to v_2 during time interval (t_1, t_2).

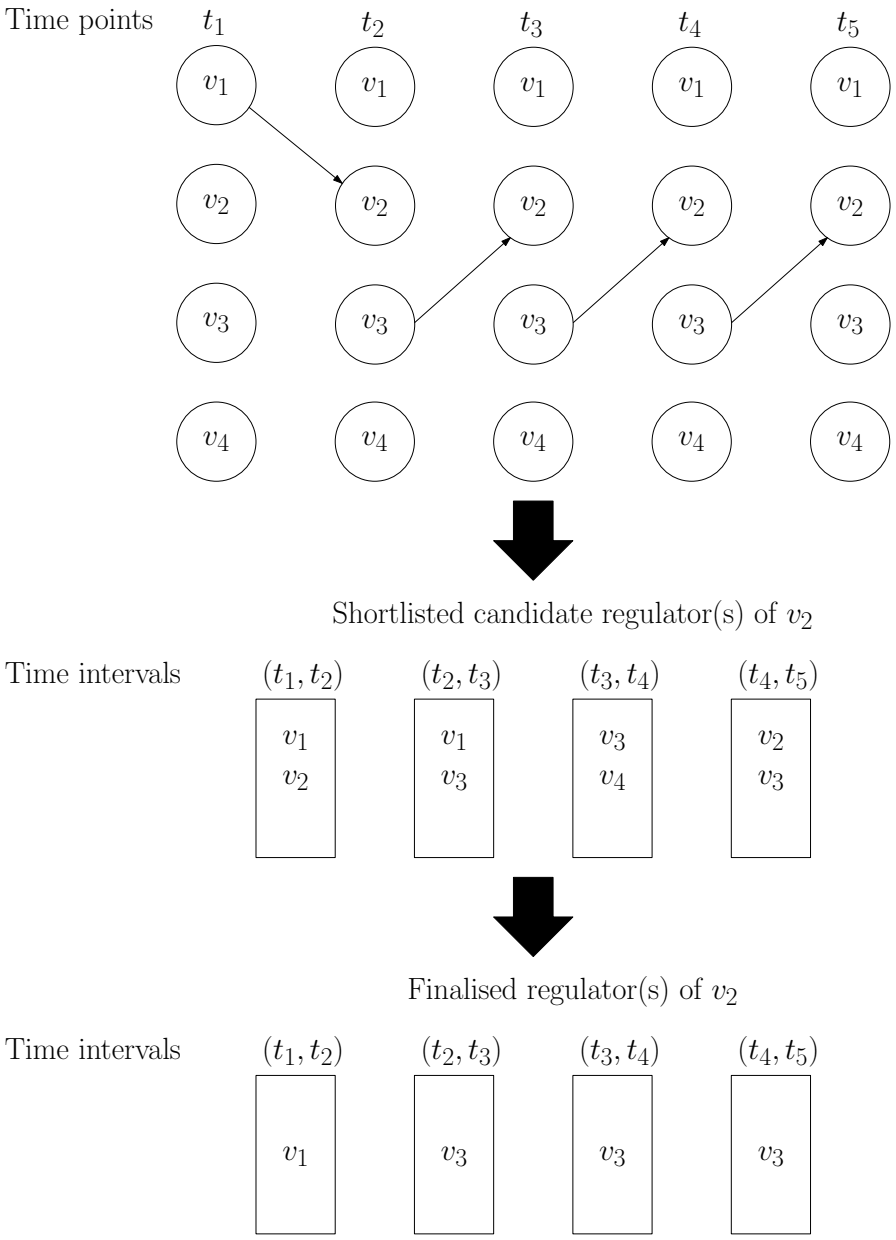


Figure 8.2: An Example for Illustrating the Idea of Time-varying Short-listing. In this example, a system with four genes – $\{v_1, \cdots, v_4\}$ – is considered. Among them, we arbitrarily choose v_2 whose regulator(s) we want to predict. The true regulator(s) of v_2 during different time intervals are shown with the directed edges, across five time points – $\{t_1, \cdots, t_5\}$. We assume that these relationships are reflected in a time-series dataset. Given that dataset, one shortlist of candidate regulators is generated for each time interval. Each shortlist is expected to capture the true regulator(s) during the corresponding time interval, such as v_1 is shortlisted during (t_1, t_2); it may also incorrectly capture some false regulator(s), like v_2 itself is shortlisted during the same time interval. For each time interval, the final selection step uses the data and shortlist, both specific to that interval, and predicts the final set of regulators. Since, the transient regulators are captured in the shortlists, they are highly likely to be captured in the final lists. Thus, the idea of time-varying short-listing is likely to capture more transient edges than that of the previous strategy of time-invariant short-listing.

Algorithm 6 An Example with the *CLR-T* Algorithm

procedure CLR-T($\mathcal{D}^*, \mathscr{M}^*$)
    ## The objective of this example is to determine whether gene v_1 should be
    ## shortlisted as a candidate regulator of gene v_2 for time-interval (t_1, t_2).
    ## Input data $\mathcal{D}^*$ represents the expression values of all genes at time-points
    ## t_1 and t_2.
    ## Input matrix $\mathscr{M}^*$ represents the mutual information matrix specific to
    ## time-interval (t_1, t_2). It is a $(V \times V)$ matrix, where V = number of genes.
    ## The $(v_i, v_j)^{th}$ cell, denoted by $\mathscr{M}^*(v_i, v_j)$, represents the estimated
    ## mutual information value between the expressions of v_i at time-point t_1 and
    ## that of v_j at time-point t_2.
    $\mu_1 \leftarrow$ arithmetic mean of $\{\mathscr{M}^*(v_i, v_2) : \text{for all } v_i\}$.
    $\mu_2 \leftarrow$ arithmetic mean of $\{\mathscr{M}^*(v_1, v_j) : \text{for all } v_j\}$.
    if $\mathscr{M}^*(v_1, v_2)$ is either greater than μ_1 or μ_2 **then**
        Shortlist v_1 as a candidate regulator of v_2 for time-interval (t_1, t_2).
    else
        Do not shortlist v_1.
    end if
end procedure

Based on the modified framework, we extend *TGS* to develop a new algorithm. This algorithm employs *CLR-T* for the short-listing step and performs the final selection step the same way *TGS* does. For that reason, we name the new algorithm 'TGS - having Time-varying shortlists' (*TGS-T*).

Similarly, we extend *TGS-Lite* to develop another new algorithm named 'TGS-T-Lite'. The latter algorithm employs *CLR-T* for the short-listing step and executes the final selection step the same way *TGS-Lite* does.

However, a problem arises when we attempt to extend *TGS+* and *TGS-Lite+* with the modified framework. We discuss its origin in the following sub-section.

8.3.1 The Issue with Extending *TGS+* and *TGS-Lite+*

In *TGS+* and *TGS-Lite+*, the short-listing step is performed based on 'refined' mutual information values. Suppose, the objective is to decide whether to shortlist gene v_i as a candidate regulator of gene v_j. In that case, their mutual information is calculated from the whole time-series data. This mutual information is called their 'raw' mutual information. If the raw mutual information is greater than zero, then an algorithm named *ARACNE* is employed to verify whether this mutual information is due to a direct relationship or an indirect one (Margolin et al., 2006). A direct relationship represents that v_i directly regulates v_j. On the other hand, an example of indirect relationships is when v_i regulates a third gene v_k which in turn regulates v_j, thus causing a non-zero mutual information between v_i and v_j. If the relationship is detected to be an indirect one, then the 'refined' mutual information between v_i and v_j is considered to be zero. Otherwise, their refined mutual information is regarded same as the raw one. Based on the refined mutual information, *CLR* decides whether to shortlist v_i as a candidate regulator of v_j.

The aforementioned strategy works because both *ARACNE* and *CLR* are time-invariant. The output of *ARACNE* is a refined mutual information matrix, which is time-invariant.

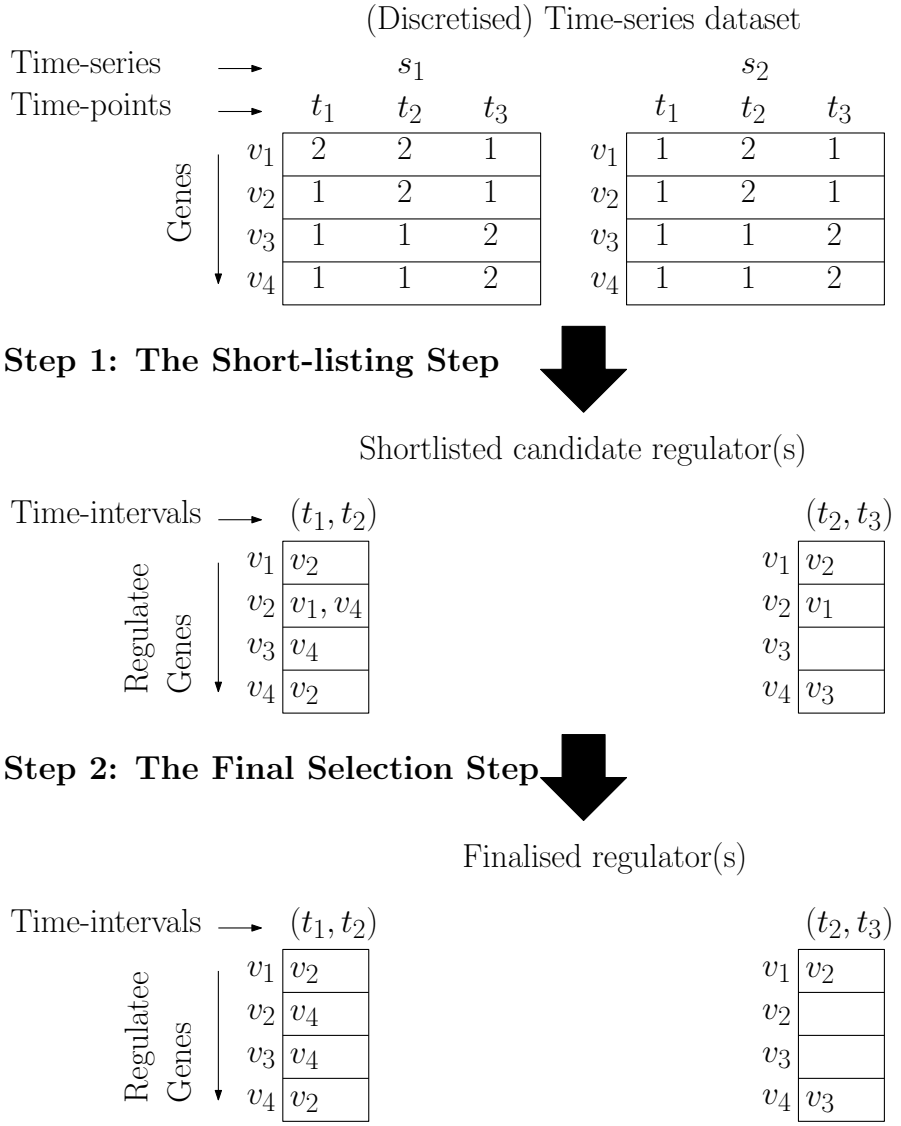


Figure 8.3: Illustration of the Modified Framework with an Example. In this example, the input dataset is comprised of two time series – s_1 and s_2. Each time series contains the expressions of four genes $\{v_1, \cdots, v_4\}$ across three time points $\{t_1, \cdots, t_3\}$. The dataset is discretised as required by the proposed framework. Given this dataset, step 1 of the modified framework generates time-varying shortlists of candidate regulators. Consequently in Step 2, the final sets of regulators are chosen from the corresponding shortlists.

On the other hand, *CLR* expects a time-invariant mutual information matrix as input. Thus, *ARACNE* and *CLR* are perfectly compatible.

However, if we wish to replace *CLR* with *CLR-T*, then the expected input is a set of time-varying mutual information matrices – one matrix for each time interval. Thus, *CLR-T* is not compatible with *ARACNE*. Therefore, we need to develop an algorithm that can produce one refined mutual information matrix for each time interval. We attempt to do that in the next sub-section.

8.3.2 Developing a Time-varying Refinement Strategy

We begin by investigating how *ARACNE* performs its refinements. Suppose, there is a non-zero mutual information between genes v_i and v_j, denoted by $\mathscr{M}(v_i, v_j)$. Then, *ARACNE* needs to determine whether it is due to a direct or an indirect relationship.

For that purpose, *ARACNE* checks whether there exists any other gene v_k $(k \neq i, k \neq j)$ such that $\mathscr{M}(v_i, v_j)$ is less than $\mathscr{M}(v_i, v_k)$ and $\mathscr{M}(v_k, v_j)$. In other words, $\mathscr{M}(v_i, v_j) < \min(\mathscr{M}(v_i, v_k), \mathscr{M}(v_k, v_j))$. If there exists at least one v_k that satisfies this condition, then *ARACNE* assumes that v_i and v_j do not share a direct relationship; instead, they share an indirect relationship via that v_k (Figure 8.4).

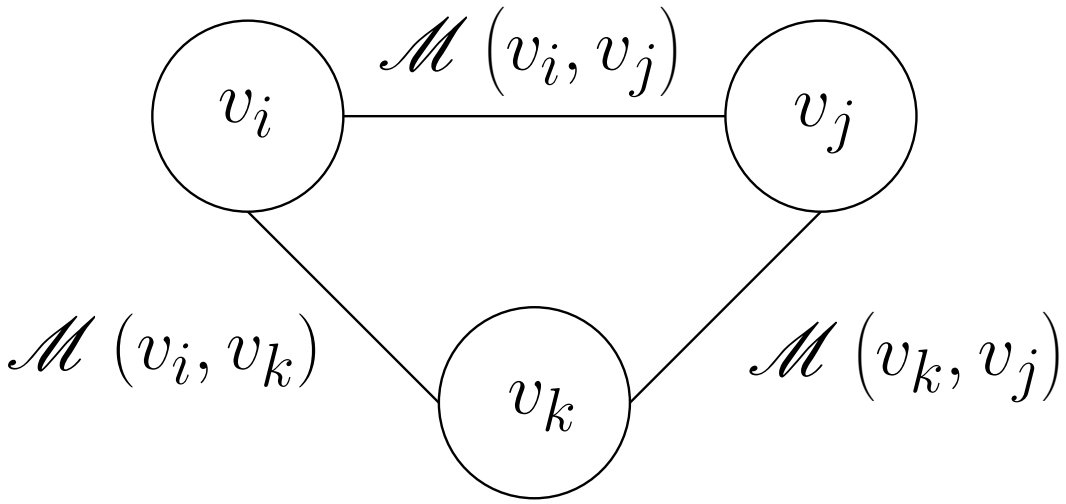
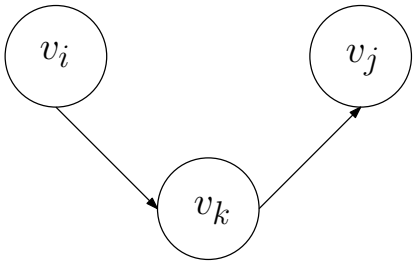


Figure 8.4: Illustration of the Time-invariant Refinement strategy by the *ARACNE* Algorithm. The strategy is explained with an example. In this example, we assume that two genes – v_i and v_j share a non-zero mutual information, denoted by $\mathscr{M}(v_i, v_j)$. It is represented with an undirected edge since mutual information is symmetric in nature i.e. $\mathscr{M}(v_i, v_j) = \mathscr{M}(v_j, v_i)$. Therefore, *ARACNE* needs to figure out whether this mutual information is due to a direct regulatory relationship between the two genes or not. For that purpose, ARACNE searches for a third gene v_k that satisfies the condition: $\mathscr{M}(v_i, v_j) < \min(\mathscr{M}(v_i, v_k), \mathscr{M}(v_k, v_j))$. If *ARACNE* is able to find at least one such v_k, it assumes that v_i and v_j do not have a direct regulatory relationship. Instead, they are related through v_k. One such case could be where v_i regulates v_k and v_k in turn regulates v_j. In such cases, *ARACNE* sets $\mathscr{M}(v_i, v_j) = 0$. On the other hand, if *ARACNE* is unable to find any such v_k, it assumes that v_i and v_j have a direct regulatory relationship. Therefore, the value of $\mathscr{M}(v_i, v_j)$ is kept unchanged. Thus, *ARACNE* checks and refines (if necessary) the mutual information values between every pair of genes.

The condition used in *ARACNE* comes from the theorem of Data Processing Inequality (DPI) (Margolin et al., 2006). The DPI states that, if genes v_i and v_j are related only through a third gene v_k, then the following inequality must hold: $\mathscr{M}(v_i, v_j) \leq \min(\mathscr{M}(v_i, v_k), \mathscr{M}(v_k, v_j))$.

Therefore, if the only relationship between v_i and v_j is that v_i regulates v_k and v_k in turn regulates v_j, then it follows from the DPI that: $\mathscr{M}(v_i, v_j) \leq \min(\mathscr{M}(v_i, v_k), \mathscr{M}(v_k, v_j))$ (Figure 8.5).



$$\mathscr{M}\left(v_i, v_j\right) \leq \min\left(\mathscr{M}\left(v_i, v_k\right), \mathscr{M}\left(v_k, v_j\right)\right)$$

Figure 8.5: An Example of the DPI. In this example, gene v_i regulates gene v_k and v_k in turn regulates gene v_j. No other regulatory relationships exist between v_i and v_j. In that case, the DPI states that the following inequality must be satisfied: $\mathscr{M}\left(v_i, v_j\right) \leq \min\left(\mathscr{M}\left(v_i, v_k\right), \mathscr{M}\left(v_k, v_j\right)\right)$.

From the DPI, it is conjectured by Margolin et al. that, if there exists such a v_k which satisfies the inequality, then it is highly likely that v_i and v_j are only indirectly related through v_k (Margolin et al., 2006). The conjecture is fundamentally the converse of the DPI. It is useful when considering whether to shortlist v_i as a candidate regulator of v_j or not. v_i is not shortlisted if the conjecture is satisfied by a third gene v_k.

However, the conjecture can not be applied directly when v_i and v_j belong to two different time points, as in the case with our modified framework. In this framework, v_i belongs to the previous time point of that of v_j. The objective is to determine whether to shortlist v_i at time point t_p (in short, $v_i\_t_p$) as a candidate regulator of v_j at time point $t_{(p+1)}$ (in short, $v_j\_t_{(p+1)}$). For that purpose, we begin by computing mutual information $\mathscr{M}\left(v_i\_t_p, v_j\_t_{p+1}\right)$. If it is zero, then we reject $v_i\_t_p$ without further examinations. Otherwise, for a non-zero mutual information, we need to examine further to identify whether it is due to an indirect relationship.

For having an indirect relationship, there must be a gene v_k that directly regulates $v_j\_t_{(p+1)}$. To be a direct regulator of $v_j\_t_{(p+1)}$, v_k must belong to time point t_p; hence, we rename it to $v_k\_t_p$ ($k \neq i$). If $v_j\_t_{p+1}$ is regulated by $v_k\_t_p$ and not by $v_i\_t_p$, then from the DPI it follows that the mutual information between $v_i\_t_p$ and $v_j\_t_{p+1}$ is not higher than that between $v_k\_t_p$ and $v_j\_t_{p+1}$. In other words, $\mathscr{M}\left(v_i\_t_p, v_j\_t_{(p+1)}\right) \leq \mathscr{M}\left(v_k\_t_p, v_j\_t_{(p+1)}\right)$.

Nevertheless, we can not reject $v_i\_t_p$ only based on the criterion that it shares less mutual information with $v_j\_t_{(p+1)}$ than $v_k\_t_p$ does. The reason is that the criterion alone does not reject the possibility of $v_i\_t_p$ and $v_k\_t_p$ together regulating $v_j\_t_{(p+1)}$.

Therefore, we need to identify a stronger criterion to claim that $v_i\_t_p$ shares a non-zero mutual information with $v_j\_t_{(p+1)}$ only due to the presence of an indirect relationship. More specifically, our claim is that $v_i\_t_p$ shares a non-zero mutual information with one of the true regulators of $v_j\_t_{(p+1)}$. Without loss of generality, let us assume that the true regulator of $v_j\_t_{(p+1)}$ in question is $v_k\_t_p$. Then, $v_i\_t_p$ shares a non-zero mutual information with $v_k\_t_p$. On the other hand, $v_k\_t_p$ shares a non-zero mutual information with its direct regulatee $v_j\_t_{(p+1)}$. As a result, there exists a non-zero mutual information between $v_i\_t_p$ and $v_j\_t_{(p+1)}$. In that case, from the DPI, we can derive the following postulate:

Postulate 1. *Let us make the following assumptions:*

- *Assumption 1. $v_i\_t_p$ does not regulate $v_j\_t_{(p+1)}$.*

- *Assumption 2.* $v_k\_t_p$ $(k \neq i)$ *regulates* $v_j\_t_{(p+1)}$.
- *Assumption 3.* $v_i\_t_p$ *shares a non-zero mutual information with* $v_j\_t_{(p+1)}$. *The only reason behind this observation is that* $v_i\_t_p$ *shares a non-zero mutual information with* $v_k\_t_p$.

Given the assumptions, it can be stated from the DPI that the mutual information between $v_i\_t_p$ *and* $v_j\_t_{(p+1)}$ *is not higher than that between* $v_i\_t_p$ *and* $v_k\_t_p$. *From the DPI, it can also be stated that the mutual information between* $v_i\_t_p$ *and* $v_j\_t_{(p+1)}$ *is not higher than that between* $v_k\_t_p$ *and* $v_j\_t_{(p+1)}$. *In other words,*
$\mathscr{M}\left(v_i\_t_p, v_j\_t_{(p+1)}\right) \leq \min\left(\mathscr{M}\left(v_i\_t_p, v_k\_t_p\right), \mathscr{M}\left(v_k\_t_p, v_j\_t_{(p+1)}\right)\right)$.

By taking the converse of Postulate 1 , we can devise the following conjecture:

Conjecture 1. *If* $\mathscr{M}\left(v_i\_t_p, v_j\_t_{(p+1)}\right) \leq \min\left(\mathscr{M}\left(v_i\_t_p, v_k\_t_p\right), \mathscr{M}\left(v_k\_t_p, v_j\_t_{(p+1)}\right)\right)$, *then*

- $v_i\_t_p$ *does not regulate have a direct relationship with* $v_j\_t_{(p+1)}$.
- $v_i\_t_p$ *has an indirect relationship with* $v_j\_t_{(p+1)}$ *only through* $v_k\_t_p$.

A realistic case where Conjecture 1 holds is shown in Figure 8.6 .

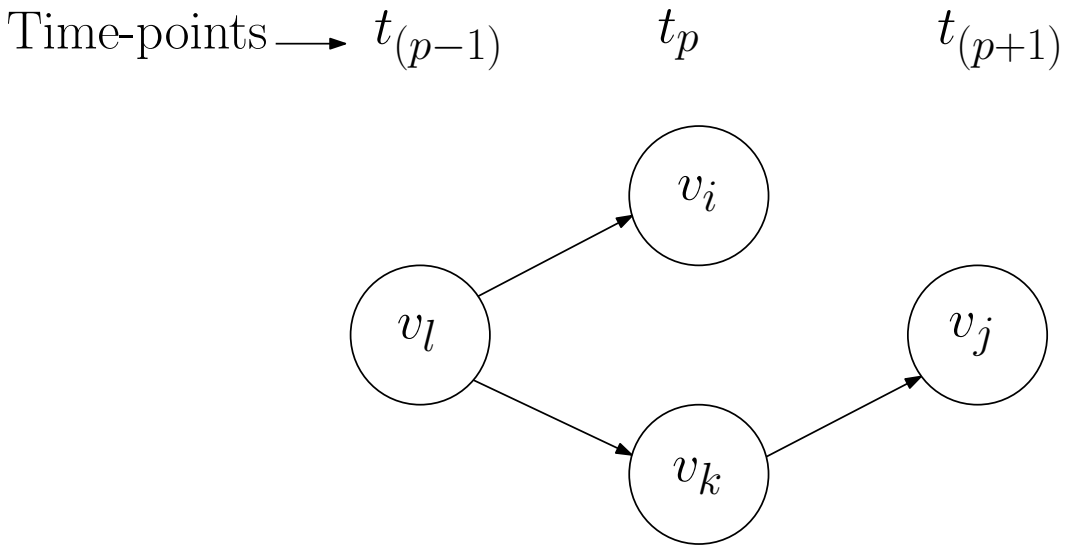


Figure 8.6: A Realistic Case where Conjecture 1 Holds. In this case, $v_i\_t_p$ and $v_k\_t_p$ are co-regulated by the same gene, suppose, $v_l\_t_{(p-1)}$. Therefore, the formers are highly likely to share a non-zero mutual information i.e. $\mathscr{M}\left(v_i\_t_p, v_k\_t_p\right) > 0$. On the other hand, $v_k\_t_p$ regulates $v_j\_t_{(p+1)}$. Hence, $v_k\_t_p$ and $v_j\_t_{(p+1)}$ also share a non-zero mutual information i.e. $\mathscr{M}\left(v_k\_t_p, v_j\_t_{(p+1)}\right) > 0$. Since $v_i\_t_p$ shares a non-zero mutual information with $v_k\_t_p$ which again shares a non-zero mutual information with $v_j\_t_{(p+1)}$, $v_i\_t_p$ might share a non-zero mutual information with $v_j\_t_{(p+1)}$ i.e. $\mathscr{M}\left(v_i\_t_p, v_j\_t_{(p+1)}\right) > 0$. However, according to the DPI, this false mutual information must not exceed the true mutual informations that have caused its non-zeroness. In other words, $\mathscr{M}\left(v_i\_t_p, v_j\_t_{(p+1)}\right)$ must not exceed $\mathscr{M}\left(v_i\_t_p, v_k\_t_p\right)$ or $\mathscr{M}\left(v_k\_t_p, v_j\_t_{(p+1)}\right)$.

Based on Conjecture 1 , we design a variant of *ARACNE*, namely - 'ARACNE - which is Time-varying'; in short, the *ARACNE-T* algorithm. It is demonstrated with an example in Algorithm 7 .

Thus, *ARACNE-T* refines every raw mutual information value. The full version of *ARACNE-T* takes an ordered list of time-varying raw mutual information matrices as input. The number of matrices in the list is equal to the number of time intervals. The p^{th} element in the list is the raw mutual information matrix specific to interval $\left(t_p, t_{(p+1)}\right)$. It is a $(V \times V)$ matrix, where V = number of genes. The $(v_i, v_j)^{th}$ cell of the matrix represents the raw mutual information between $v_i\_t_p$ and $v_j\_t_{(p+1)}$. For every cell in each matrix, *ARACNE-T* reads the raw value and replaces it with the refined value. As a consequence, the output of *ARACNE-T* is an ordered list of time-varying refined mutual information matrices. We illustrate the workflow of *ARACNE-T* with an example in Figure 8.7 .

Algorithm 7 An Example with the *ARACNE-T* Algorithm

procedure ARACNE-T($\mathscr{M}\left(v_i\_t_p, v_j\_t_{(p+1)}\right)$, $\mathcal{D}^*$)
    ## Input $\mathscr{M}\left(v_i\_t_p, v_j\_t_{(p+1)}\right)$ represents the 'raw' mutual information between
    ## $v_i\_t_p$ and $v_j\_t_{(p+1)}$ as estimated from data $\mathcal{D}^*$.
    ## $\mathcal{D}^*$ represents the gene expression data at time points t_p and $t_{(p+1)}$.
    ## Assuming that the raw mutual information is non-zero,
    ## the objective for this example is to determine whether the non-zeroness
    ## is due to a direct relationship between $v_i\_t_p$ and $v_j\_t_p$ or an indirect one.

    if there exists at least one $v_k\_t_p$ $(k \neq i)$ that
        satisfies the following condition:
        $\mathscr{M}\left(v_i\_t_p, v_j\_t_{(p+1)}\right) < \min\left(\mathscr{M}\left(v_i\_t_p, v_k\_t_p\right), \mathscr{M}\left(v_k\_t_p, v_j\_t_{(p+1)}\right)\right)$ **then**
        The non-zeroness is due to an indirect relationship only through $v_k\_t_p$.
        Therefore, the true mutual information between $v_i\_t_p$ and $v_j\_t_{(p+1)}$
        should be zero.
        Set $\mathscr{M}\left(v_i\_t_p, v_j\_t_{(p+1)}\right) \leftarrow$ zero.
        ## $\mathscr{M}\left(v_i\_t_p, v_k\_t_p\right)$ and $\mathscr{M}\left(v_k\_t_p, v_j\_t_{(p+1)}\right)$ are computed from $\mathcal{D}^*$.
    else
        The non-zeroness is due to a direct relationship.
        Keep $\mathscr{M}\left(v_i\_t_p, v_j\_t_{(p+1)}\right)$ unchanged.
    end if

    ## At the end of the procedure,
    ## the value of $\mathscr{M}\left(v_i\_t_p, v_j\_t_{(p+1)}\right)$ is called
    ## the 'refined' mutual information between $v_i\_t_p$ and $v_j\_t_{(p+1)}$.
end procedure

Therefore, the output of *ARACNE-T* is compatible with the input of *CLR-T*, which expects a set of mutual information matrices. As a consequence, we can replace *ARACNE* with *ARACNE-T*, and *CLR* with *CLR-T* in the TGS+ algorithm (Figure 8.8). Hence, a new algorithm is developed by extending *TGS+* with the aforementioned replacements. We name this algorithm – *TGS-T+*.

Similarly, we extend *TGS-Lite+* by performing the same replacements and develop another new algorithm. We name this algorithm – *TGS-T-Lite+*.

8.3.3 Section Summary

In this section, we propose four novel algorithms. They are – *TGS-T*, *TGS-T+*, *TGS-T-Lite* and *TGS-T-Lite+*. Each of these algorithms is an extension of one of the algorithms

proposed in the previous chapters. Each extended algorithm is designed to capture more transient edges than that of the original one.

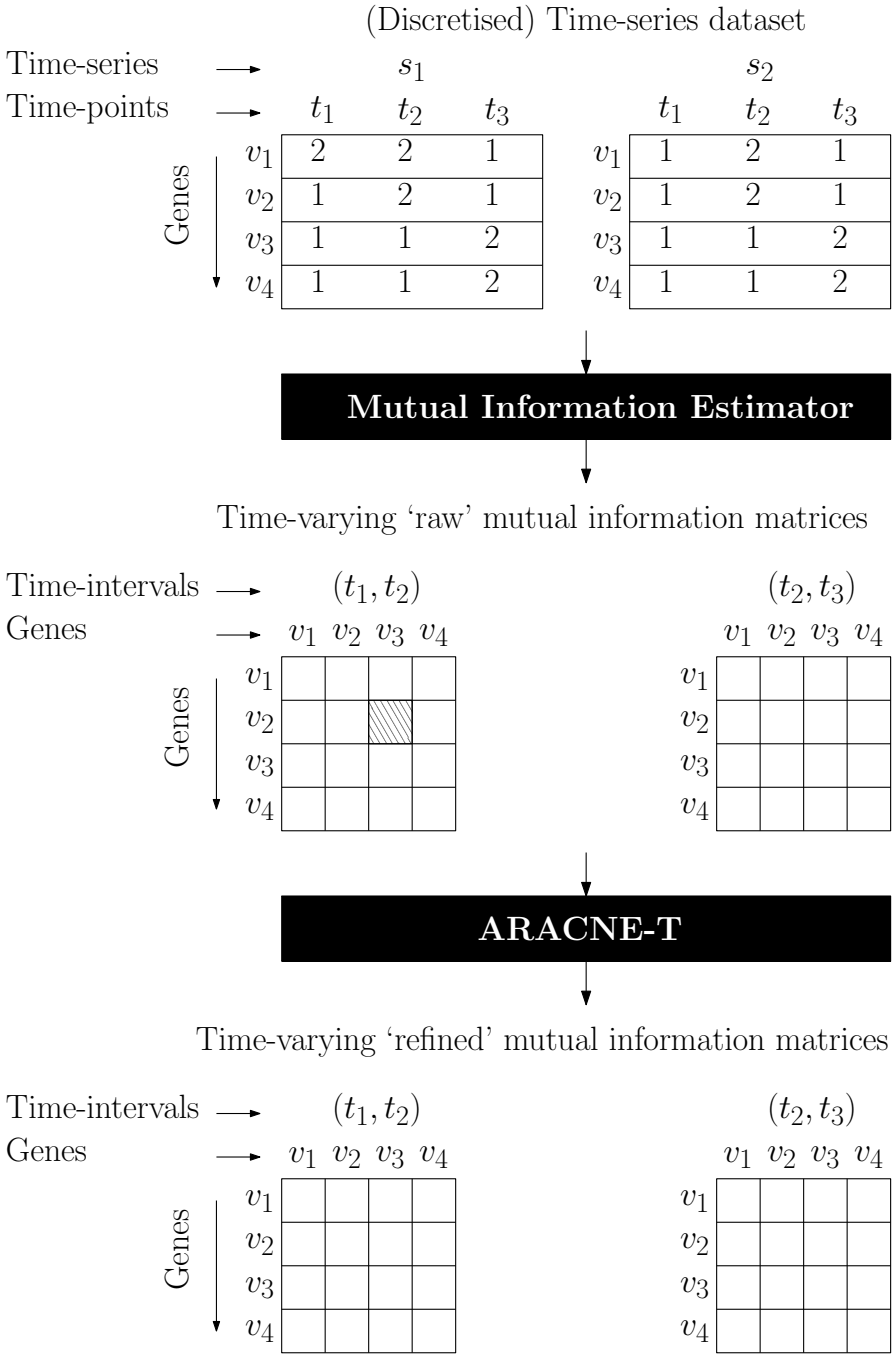


Figure 8.7: Illustration of the Workflow of *ARACNE-T* with an Example. In this example, the input dataset is disretised and comprised of two time series – s_1 and s_2. Each time series contains the expressions of four genes $\{v_1, \cdots, v_4\}$ across three time points $\{t_1, \cdots, t_3\}$. From this dataset, an ordered list of time-varying raw mutual information matrices is estimated. Each cell in a matrix contains a non-negative real number which represents a mutual information value (the numbers are not shown inside the cells to save space). For example, the shaded cell contains the mutual information between $v_2\_t_1$ and $v_3\_t_2$. This ordered list is given as input to *ARACNE-T* which in turn produces an equivalent list of refined mutual information matrices.

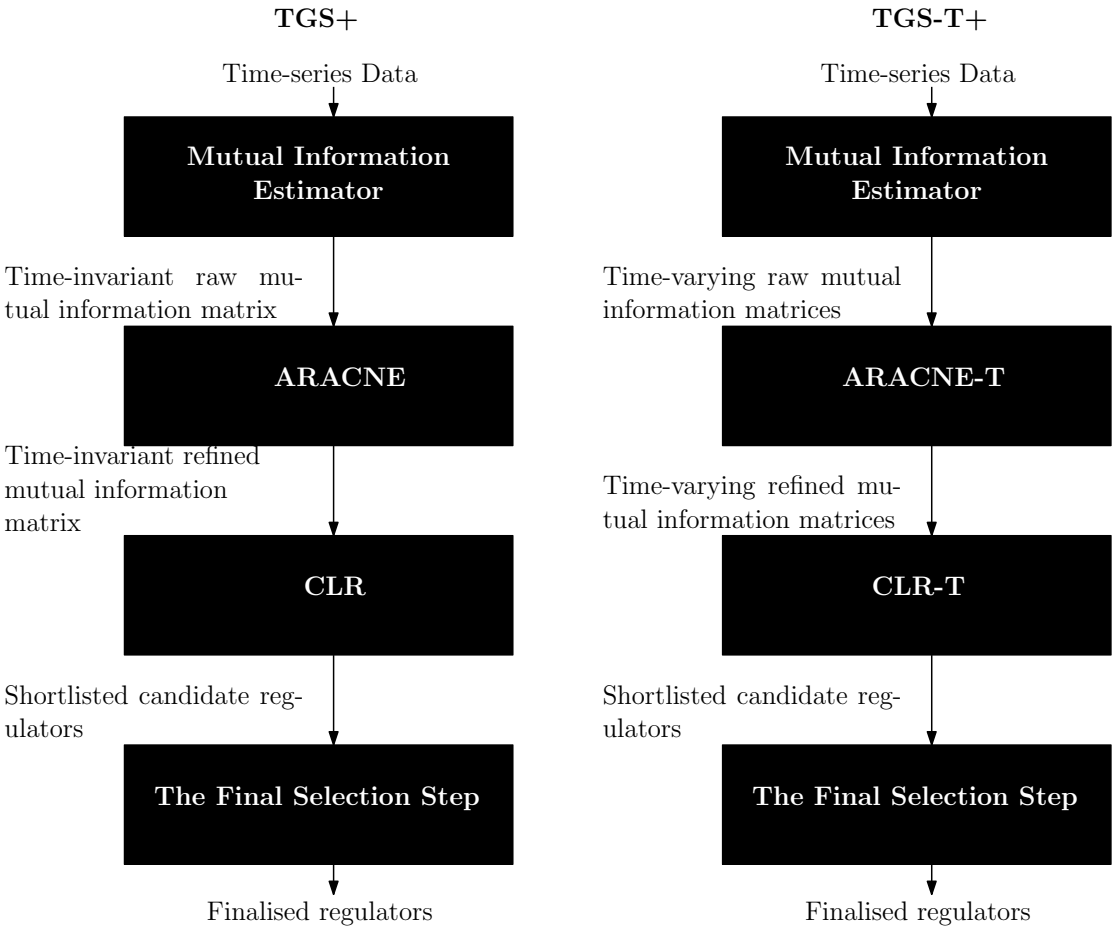


Figure 8.8: The Black Box Diagrams of Algorithms *TGS+* and *TGS-T+*. The latter (in right) differs from the former (in left) in two places: first, ARACNE is replaced with ARACNE-T; second, CLR is replaced with CLR-T.

8.4 Experimental Setup

8.4.1 Evaluation Strategy

In the previous section, we introduce four algorithms. They are extended versions of the four algorithms we proposed in the previous chapters (Table 8.1).

Table 8.1: Mapping from the Original Algorithms to the Extended Algorithms.

Original Algorithm	**Extended Algorithm**
TGS	TGS-T
TGS+	TGS-T+
TGS-Lite	TGS-T-Lite
TGS-Lite+	TGS-T-Lite+

Each extended algorithm is expected to capture a higher number of transient edges than that of the corresponding original algorithm. Therefore, the evaluation strategy is to test whether the extended algorithms can meet this expectation when put into

competition with the original ones.

8.4.2 Implementations

In this study, we are interested in comparing the concerned algorithms with respect to correctness. In that regard, *TGS-T* and *TGS-T-Lite* should not vary since the latter is simply a more memory-efficient variant of the former. Hence, we choose to implement only one of them, which is *TGS-T*. For the same reason, we choose to implement only one among *TGS-T+* and *TGS-T-Lite+*. In this case, *TGS-T+* is selected for implementation. The implementations are done in the R programming language (R Development Core Team, 2008) version 3.5.1. In the previous chapters, we use an older version of R (version 3.3.2). While that can have effects on runtime and memory utilisations, it should not affect correctness. Therefore, the results of the alternative algorithms reported in the previous chapters are comparable to that of the proposed algorithms.

8.5 Results and Discussions

In this section, we aim to examine the results of *TGS-T* and *TGS-T+* compared to that of the alternative algorithms.

8.5.1 Comparative Study Against Alternative Algorithms

TGS-T and *TGS-T+* require discretised data. For a fair comparison, we use the same data-discretisation algorithm (namely *2L.wt*) that is used for previously proposed algorithms.

8.5.1.1 Comparison on Dataset Ds10n

For dataset Ds10n, both *TGS-T* and *TGS-T+* reconstruct null networks i.e. networks with no edges (Table 8.2) [1] . It means that none of the genes have any regulators.

Therefore, no regulators are chosen in the final selection step for every (regulatee) gene. It makes us suspect whether any regulators are chosen in the short-listing step. However, the suspicion is proved to be wrong after investigation. Many of the genes have non-empty shortlist of candidate regulators. However, none of these candidates makes it to the final list. Therefore, the question is why the short-listing step is unable to identify even a single candidate that can pass the rigour of the final selection step.

The answer is the scarcity of data. In case of *TGS* and *TGS+*, the short-listing step utilises the whole dataset; this dataset contains 21 time points and 4 instances per time point for each gene's expression. Therefore, the short-listing is performed with $(21 \times 4) = 84$ instances of every gene's expression. For example, in order to determine whether to short-list gene v_i as a candidate regulator of gene v_j, the short-listing step compares all 84 instances of v_i with all 84 instances of v_j.

On the other hand, in case of *TGS-T* and *TGS-T+*, the short-listing step generates one shortlist for each time interval, by utilising the data specific to that time interval. For example, in order to decide whether gene v_i should be short-listed as a candidate regulator of gene v_j during time-interval $\left(t_p, t_{(p+1)}\right)$, the short-listing step compares the 4 instances of v_i at time-point t_p with the 4 instances of v_j at time-point $t_{(p+1)}$.

[1] Please note that algorithms *TVDBN-exp-hard* and *TVDBN-exp-soft* result in error for Ds10n.

Thus, scarcity of instances per time point is the reason behind the poor performance of the short-listing step. Let us illustrate the point with gene G_4. In case of *TGS*, all 84 instances of G_4 is compared with that of the other genes. Consequently, genes $\{G_2, G_3, G_5, G_6, G_7, G_9, G_{10}\}$ are short-listed as candidate regulators of G_4. Among them, G_6 is correctly finalised to be a regulator of G_4. However, *TGS-T* misses this true regulator. For example, in the first time interval, the short-listing step compares the 4 instances of G_4 at the second time point with the 4 instances of every genes at the first time point. Misled by such a small number of instances, it does not short-list G_6, and instead short-lists G_5 and G_7. None of these short-listed candidates are true regulators of G_4. Both of them are correctly rejected in the final selection step. As a result, G_4 is predicted to have no regulators in the first time interval.

In summary, *TGS-T* and *TGS-T+* can not capture any true edges from dataset Ds10n due to too few instances per time point. We proceed to examine whether they can overcome this issue for dataset Ds50n that has a larger number of instances per time point.

Table 8.2: Performances of the Selected Algorithms on Dataset Ds10n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced.

Algorithm	**TP**	**FP**	**Recall**	**Precision**	**F1**
TGS-T	0	**0**	0	0	0
TGS-T+	0	**0**	0	0	0
TGS	**3**	10	**0.3**	0.231	0.261
TGS+	**3**	1	**0.3**	**0.75**	**0.429**
TGS-Lite	**3**	10	**0.3**	0.231	0.261
TGS-Lite+	**3**	1	**0.3**	**0.75**	**0.429**
ARTIVA	0	9	0	0	0
TVDBN-0	0	1	0	0	0
TVDBN-bino-hard	1	7	0.1	0.125	0.111
TVDBN-bino-soft	2	9	0.2	0.182	0.190

8.5.1.2 Comparison on Dataset Ds50n

For dataset Ds50n, *TGS-T* and *TGS-T+* are able to reconstruct non-null networks (Table 8.3) [2] .

Moreover, *TGS-T* and *TGS-T+* capture the largest number of true edges (Column 'TP' in Table 8.3). This observation is in sharp contrast with that for dataset Ds10n where these algorithms are unable to capture even a single true edge. The dissimilarity is caused by the difference between the numbers of instances among the said datasets. While Ds10n contains only 4 instances per time point for each gene's expression, Ds50n contains 23 of them.

[2]Please note that algorithms *TVDBN-exp-hard* and *TVDBN-exp-soft* result in error for Ds50n.

Table 8.3: Performances of the Selected Algorithms on Dataset Ds50n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced.

Algorithm	**TP**	**FP**	**Recall**	**Precision**	**F1**
TGS-T	**30**	632	**0.39**	0.045	0.081
TGS-T+	**30**	630	**0.39**	0.045	0.081
TGS	15	342	0.195	0.042	0.069
TGS+	6	100	0.078	0.057	0.066
TGS-Lite	15	342	0.195	0.042	0.069
TGS-Lite+	6	100	0.078	0.057	0.066
ARTIVA	6	**64**	0.078	**0.086**	**0.082**
TVDBN-0	7	199	0.091	0.034	0.049
TVDBN-bino-hard	11	410	0.143	0.026	0.044
TVDBN-bino-soft	14	395	0.182	0.034	0.058

However, we also need to consider the ratio of number of instances to number of genes in each dataset. For Ds10n, the ratio is (4 instances : 10 genes) = (4 : 10). On the other hand, this ratio for Ds50n is (23 instances : 50 genes) = (4.6 : 10). Therefore, the ratio is slightly higher for Ds50n.

Then the question arises: what exactly cause the poor performance for Ds10n? Is it the scarcity of absolute number of instances, or is it the low ratio of instances to genes, or is it both? We look forward for the comparative study with dataset Ds100n to answer this question.

At present, we move on from 'TP' to examine the performances of *TGS-T* and *TGS-T+* with respect to other metrics (Table 8.3). These two algorithms achieve the highest recall due to their high TP (Columns 'Recall' and 'TP' in Table 8.3). Nevertheless, they concede the highest numbers of false positives (Column 'FP' in Table 8.3). It also affects their precisions where they place after {*ARTIVA*, *TGS+*, *TGS-Lite+*} (Column 'Precision' in Table 8.3). In spite of that, they achieve the second highest F1-scores, less only than that of *ARTIVA* (Column 'F1' in Table 8.3). Thus, they over-compensate for their lower precisions with significantly higher recalls, compared to that of {*TGS+*, *TGS-Lite+*}.

Finally, we compare the performances of *TGS-T* and *TGS-T+* themselves, in order to study the effect of the ARACNE-T sub-step. This sub-step is present in *TGS-T+* only. It is observed that ARACNE-T helps *TGS-T+* to incur a smaller number of false positive predictions than that of *TGS-T* (Column 'FP' in Table 8.3). This observation provides a proof of concept for ARACNE-T's ability to identify and reject indirect regulatory relationships. Moreover, it does not make any sacrifices in capturing the direct regulatory relationships; as a result *TGS-T+* retains the same number of true positive predictions as that of *TGS-T* (Column 'TP' in Table 8.3). That, in turn, causes *TGS-T+* to exceed *TGS-T* with regard to precision (by $\simeq 0.00014$) and F1-score (by $\simeq 0.00022$); the differences do not get reflected in Columns 'Precision' and 'F1' of Table 8.3 because the values in the table are rounded off to three decimal places. These observations provide an exemplary evidence in support of Conjecture 1 that we introduce for designing ARACNE-T (Sub-section 8.3.2).

8.5.1.3 Comparison on Dataset Ds100n

For dataset Ds100n, *TGS-T* and *TGS-T+* are able to reconstruct non-null networks again (Table 8.4) [3] .

Table 8.4: Performances of the Selected Algorithms on Dataset Ds100n. TP = True Positive, FP = False Positive. The numerical values are rounded off to three decimal places. For each column, the best value(s) is boldfaced.

Algorithm	**TP**	**FP**	**Recall**	**Precision**	**F1**
TGS-T	**49**	1972	**0.295**	0.024	0.045
TGS-T+	**49**	1972	**0.295**	0.024	0.045
TGS	28	790	0.169	0.034	0.057
TGS+	19	181	0.114	**0.095**	**0.104**
TGS-Lite	28	790	0.169	0.034	0.057
TGS-Lite+	19	181	0.114	**0.095**	**0.104**
ARTIVA	14	**158**	0.084	0.081	0.083
TVDBN-0	9	678	0.054	0.013	0.021
TVDBN-bino-hard	26	1304	0.157	0.020	0.035
TVDBN-bino-soft	18	1296	0.108	0.014	0.024

Moreover, *TGS-T* and *TGS-T+* are able to capture the largest number of true edges (Column 'TP' in Table 8.4), like they do for Ds50n. At this point, we return to the unresolved question of why are the aforementioned algorithms unable to capture even a single true edge in case of Ds10n. Earlier, we narrow down the reasons to two: the scarcity of absolute number of instances and the low ratio of instances to genes. However, it is not resolved – which one of them, or whether both of them are responsible for the poor performance.

Earlier, we remain unable to resolve the question is because Ds50n has a larger number of instances as well as a larger ratio of instances to genes than that of Ds10n. Unfortunately, Ds100n also has a larger number of instances (46) and a larger ratio of instances to genes (4.6 : 10) than that of Ds10n (4 instances and a ratio of 4 : 10, respectively). Therefore, this question can not be resolved with the datasets at hand.

To resolve the question, we need datasets with the following properties: To test whether the absolute number of instances cause the issue, we require datasets with the same number of genes as Ds10n but with different number of instances than that of Ds10n; on the other hand, to test whether the ratio of instances to genes is the issue, we require datasets with lower ratios than that of Ds10n. This investigation is out-of-scope for the current thesis. Hence, we present this opportunity as a foundation for future works.

At present, we move on to examine the performances of *TGS-T* and *TGS-T+* with respect to the rest of the metrics (Table 8.4). Owing to their high TP, these two algorithms achieve the highest recall (Columns 'TP' and 'Recall' in Table 8.4). Nonetheless, they incur the highest number of false positives (Column 'FP' in Table 8.4). It severely affects their precisions where they rank after {*ARTIVA*, *TGS*, *TGS+*, *TGS-Lite*, *TGS-Lite+*} (Column 'Precision' in Table 8.4). Unlike in the case of Ds50n, the superiority

[3]Please note that algorithms *TVDBN-exp-hard* and *TVDBN-exp-soft* result in error for Ds100n.

in recalls does not compensate for the low precisions; as a result, *TGS-T* and *TGS-T+* obtain the same rank in F1-score as they do in precision (Column 'F1' in Table 8.4).

Lastly, we compare the performance of *TGS-T* with that of *TGS-T+*. It is observed that *TGS-T+* makes the same number of false positive predictions as that of *TGS-T* (Column 'FP' in Table 8.4). From this observation, we conclude that the ARACNE-T sub-step in *TGS-T+* can not identify any additional false edges. On the other hand, it is also observed that *TGS-T+* is able to make the same number of true positive predictions as that of *TGS-T* (Column 'TP' in Table 8.4). Therefore, the ARACNE-T sub-step does not miss any true edges. These observations are slightly different from that of Ds50n. In case of Ds50n, the ARACNE-T sub-step identifies two additional false edges without missing any true edges. Thus, from the combined observations of Ds50n and Ds100n, we deduce that the ARACNE-T sub-step helps *TGS-T+* to reject monotonically higher numbers of false edges than that of *TGS-T*, while capturing the same number of true edges as that of the latter.

With the aforementioned deduction, we conclude the comparatice study of the proposed algorithms against alternative algorithms. However, it does not guarantee that the performances of the proposed algorithms are better than that of a random classifier. Hence, we conduct an additional study which is presented in the following sub-section.

8.5.2 Additional Comparative Study Against a Random Classifier

A random classifier is an algorithm that randomly decides whether an edge should be present or absent in the predicted network. Predictions of a random classifier tend to result in a line represented by equation 'True Positive Rate = False Positive Rate' (TPR = FPR); here, TPR = Recall; FPR = (FP / (FP + TN)) (Liu et al., 2016). We plot the TPR-vs-FPR line of a random classifier against that of *TGS-T* and *TGS-T+* (Figure 8.9).

From the plots, we make the following observations. For Ds10n, *TGS-T* and *TGS-T+* reconstruct null networks. Hence, their line overlaps with that of the random classifier. However, for Ds50n and Ds100n, where *TGS-T* and *TGS-T+* decide to reconstruct non-null networks, their line stays above that of the random classifier. It signifies that, when *TGS-T* and *TGS-T+* decide to reconstruct non-null networks, they make better decisions than randomly made decisions.

8.6 Excerpt and Future Work

In this chapter, we address an issue encountered by the algorithms proposed in the previous chapters. The issue corresponds to their inability to capture transient edges – the edges that remain active for short periods of time. Capturing such edges is crucial for understanding the underlying mechanisms since they can trigger long-lasting downstream effects.

To overcome the aforementioned issue, we propose a novel algorithm named *TGS-T*. A comparative study is conducted between *TGS-T* and the previously proposed algorithms. For two of the three benchmark datasets, *TGS-T* captures significantly more edges than that of the previously proposed algorithms.

However, *TGS-T* makes more false-positive predictions than that of the previous algorithms. To prevent that, we propose another algorithm named *TGS-T+*.

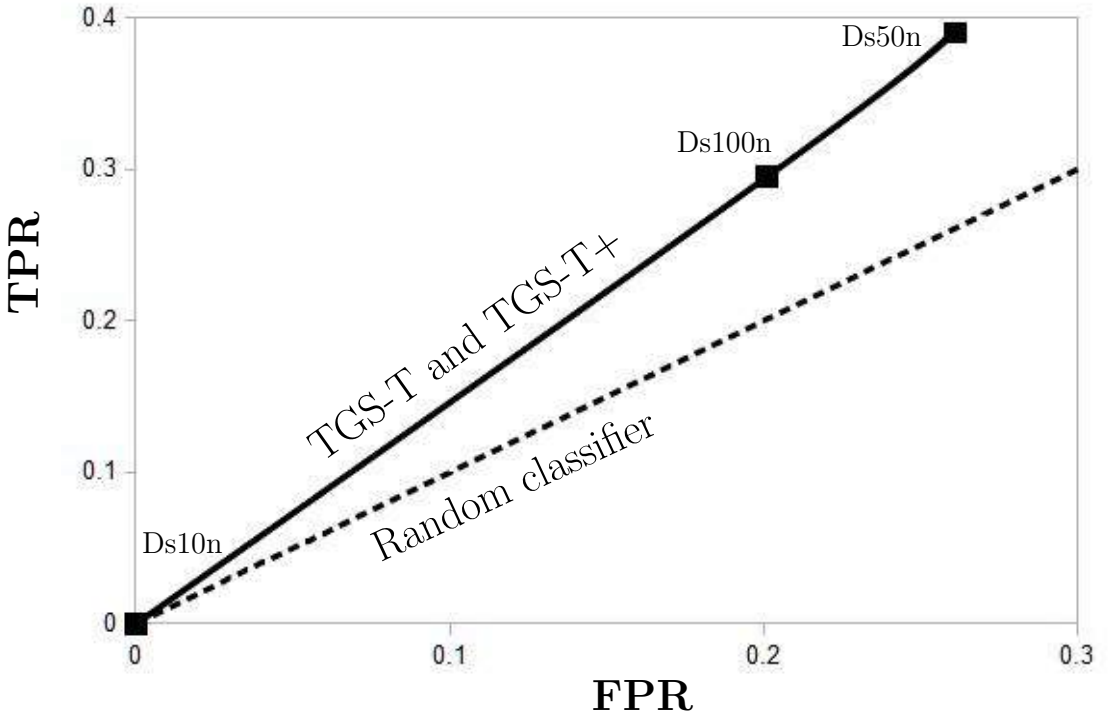


Figure 8.9: The TPR-vs-FPR Plots of *TGS-T*, *TGS-T+* and a random classifier. Here, TPR = True Positive Rate, FPR = False Positive Rate. The results of *TGS-T* and *TGS-T+* for three benchmark datasets – Ds10n, Ds50n and Ds100n – are represented as three black squares. These squares are connected (interpolated) with a smooth line (Software used: LibreOffice Calc Version 5.4.7.2 (x64); Line type = Cubic spline, Resolution = 20; OS: Windows 10 Pro version 1809). On the other hand, the results of the random classifier is presented as a dashed line.

TGS-T+ algorithm has an additional step that attempts to eliminate false edges. We observe that *TGS-T+* produces monotonically fewer numbers of false positives than that of *TGS-T*. At the same time, the former captures as many true edges as that of the latter algorithm.

Although *TGS-T+* makes fewer numbers of false-positive predictions than that of *TGS-T*, the numbers remain higher than that of the previously proposed algorithms. As a result, *TGS-T+* can not consistently outperform the previously proposed algorithms in F1-score, for all benchmark datasets. Developing algorithms that can supersede the previously proposed algorithms in F1-score remains an opportunity for the future.

8.7 Contributions

A manuscript of this work, titled "Capturing Transient Edges in Time-varying Gene Regulatory Networks", is in preparation.

8.8 Chapter Summary

In the previous chapter, we propose a set of reconstruction algorithms (Chapter 7). The set contains two algorithms: *TGS-Lite* and *TGS-Lite+*. Among them, *TGS-Lite+* provides correctness competitive to that of *ARTIVA*. At the same time, the former offers

time- and memory-efficiency compatible with large-scale datasets. However, *TGS-Lite+* shares a common limitation with the other previously-proposed algorithms. All of them tend to miss the edges that remain active for a small number of time intervals. Such edges, known as 'transient edges', can have crucial down-stream effects. Therefore, capturing them can be critical for understanding the underlying regulatory mechanisms. To overcome this limitation, we propose another set of algorithms in this chapter. Each algorithm in the set is a variant of a previously proposed algorithm. We demonstrate that every variant outperforms its original algorithm in capturing the true edges.

Chapter 9

Conclusions and Future Directions

Complex biological processes are dynamic. They progress over time through intertwined levels of biological regulations. One key level is that of gene regulation. At this level, genes regulate each other, leading to modulations in their expressions. This thesis focuses on the development of efficient algorithms that are used to reconstruct time-varying gene regulatory networks. Input to the concerned algorithms consists of time-course measurements of gene expressions.

Prior to this thesis, a number of such algorithms existed. In this thesis, we conduct a comparative study among those algorithms. It is found that an algorithm named 'Auto Regressive TIme VArying models', in short *ARTIVA*, offers the state-of-the-art correctness among the existing algorithms (Lèbre et al., 2010). However, it lacks in computational speed. As a result, *ARTIVA*'s runtime is prohibitive for processing large-scale datasets with hundreds to thousands of genes. At the same time, due to rapid advancements in data-acquisition technologies, datasets have been increasingly becoming larger. Consequently, the gap between the desired computational speed and that of *ARTIVA* has been becoming wider. In the current thesis, we bridge this gap by contributing a novel set of algorithms. They offer correctness competitive to that of *ARTIVA*, at computational speeds extremely suitable for processing large-scale datasets. A summary of the proposed algorithms is presented in the following section.

9.1 Summary of the Contributions

Improving Time-efficiency: The first algorithm proposed in this thesis is named 'an algorithm for reconstructing Time-varying Gene regulatory networks with Shortlisted candidate regulators', in short, *TGS*. This algorithm achieves significantly lower runtime than that of *ARTIVA*. However, *TGS* is unable to outperform *ARTIVA* in correctness. The correctness is measured with a widely-used metric called 'F1-score' (Liu et al., 2016). It is the harmonic mean of two other metrics called 'recall' and 'precision'. Recall measures how good an algorithm is in capturing the correct edges. On the other hand, precision measures how good an algorithm is in rejecting the incorrect edges. Therefore, F1-score is a measure of how good an algorithm is in balancing recall and precision. *TGS* is able to outperform *ARTIVA* in recall. However, the latter maintains a considerable lead over the former in precision. As a consequence, *TGS* is unable to obtain F1-scores competitive to that of *ARTIVA*.

Balancing Recall and Precision: To enhance precision, the second algorithm is

proposed. It is named 'TGS-Plus' (*TGS+*). This algorithm supersedes *ARTIVA* in correctness. Additionally, *TGS+* outpaces *TGS* in runtime. Nevertheless, it is observed that both *TGS* and *TGS+* are unable to manage their memory requirements efficiently. The requirements grow exponentially with the number of genes in a given dataset. Hence, these algorithms are expected to encounter memory-overflow issues for large-scale datasets.

Improving Memory-efficiency: To prevent memory-overflow issues, the third set of algorithms is proposed. The set contains a pair of algorithms. The first algorithm is called 'TGS - which is Light on memory', in short, *TGS-Lite*. This algorithm offers the same correctness and time-efficiency as that of *TGS*, yet, its memory requirement grows only linearly with the number of genes. Similarly, the second algorithm, known as 'TGS-Lite Plus' (*TGS-Lite+*), delivers the same correctness and time-efficiency as that of *TGS+*, at a linear memory requirement. Nonetheless, we observe that these algorithms, along with the ones we proposed earlier, tend to fail in capturing the edges that remain active for short periods of time. Such edges, known as 'transient edges', may trigger long-lasting cascading effects. Therefore, capturing transient edges is crucial for understanding the underlying gene-regulation process.

Capturing Transient Edges: For capturing transient edges, the fourth and final set of algorithms is proposed. This set consists of four algorithms. The first algorithm is called 'TGS - having Time-varying shortlists', in short, *TGS-T*. This algorithm captures significantly more numbers of correct edges than that of *TGS*. Similarly, the last three algorithms, named as {*TGS-T+*, *TGS-T-Lite*, *TGS-T-Lite+*} capture considerably more numbers of correct edges than that of {*TGS+*, *TGS-Lite*, *TGS-Lite+*}, respectively.

Nevertheless, there remain scopes for advancements. We discuss a few of them in the following section.

9.2 Limitations and Future Directions

Methodological Improvements

Improvement of Precision: As discussed in the last section, the fourth set of algorithms are able to capture considerably more numbers of correct edges than that of the previously proposed algorithms. As a result, the former algorithms significantly outperform the latter algorithms in 'recall', which is a metric for measuring how good an algorithm is in capturing the correct edges (Liu et al., 2016). However, the former algorithms are unable to outperform the latter algorithms in 'precision', which is a metric for measuring how good an algorithm is in rejecting the incorrect edges (Liu et al., 2016). Therefore, future research can consider the challenge of developing algorithms that offer recalls competitive to that of the former algorithms and precisions competitive to that of latter algorithms.

Improvement of Recall: Prior to this thesis, the highest recall achieved by any algorithm for the three benchmark datasets was 0.3. The fourth set of algorithms proposed in this thesis advance that to 0.39. Nonetheless, there are miles to go before we reach the perfect recall of 1. Therefore, future research can be channelised in that direction.

Accommodation of Continuous Data: The algorithms proposed in this thesis require discrete data as input. Since the benchmark datasets contain continuous data,

they are discretised with a conventional data-discretisation method. The discretisation process is highly likely to have caused a loss of information. As contemporary datasets usually contain continuous data, future research can be focused on developing algorithms that are at least as correct and efficient as the proposed algorithms, and can accommodate continuous data (Zaas et al., 2009).

Benchmark Creation

In this thesis, we utilise three benchmark datasets. They were published by an online competition organised by the 'DREAM Challenges' community (DREAM3). The word 'DREAM' stands for 'Dialogue on Reverse Engineering and Assessment Methods'. As the name suggests, the competition presented the datasets for assessing reverse-engineering methods. Since then, the datasets have been widely used for benchmarking reconstruction algorithms; the publications related to the creation of these datasets have received over a thousand citations (Marbach et al., 2010, 2009; Prill et al., 2010).

Nevertheless, these benchmarks have limitations. Each of the benchmarks contains a time-series dataset. However, the true time-varying gene regulatory networks that were used to generate the dataset are not made available with the corresponding benchmark.

Instead, a single time-invariant network is made available. This network represents the edge-wise union of the true time-varying networks. For example, suppose that there are two true time-varying networks; the true network specific to the first time interval contains only one edge, from gene A to gene B; the true network specific to the second time interval also contains only one edge, from gene B to gene C. In that case, the true time-varying networks are represented by a single time-invariant network that has exactly two edges, A to B, and B to C.

Therefore, the temporal sequence of the true edges is not known. As a result, an algorithm that captures the true edges in a wrong sequence would be considered as correct as an algorithm that captures them in the correct sequence. Following the previous example, if an algorithm predicts edge B to C in the first time interval and edge A to B in the second time interval, then also the prediction would be considered correct.

To overcome this limitation, future research can be conducted to create benchmarks that include time-series datasets and the corresponding true time-varying gene regulatory networks.

Utilisation of Multi-omics Data

For predicting the 'regulator $\rightarrow$ regulatee' relationships between genes, the proposed algorithms solely utilise gene expression data. This data represents only one type of omics data, which is the 'transcriptomics' data. The algorithms can be extended to utilise complementary omics datasets to improve their predictions (Kim et al., 2014). For instance, the 'functional genomics' data can be used as prior knowledge to reduce the false positive predictions. This data contains information about 'transcription factor (TF) binding sites'; TFs are protein molecules that the regulator gene sends to the regulatee gene; after reaching the regulatee gene, the TF physically binds to a location known as the binding site of the regulatee gene; upon binding, the TF produces the desired regulatory effect on the regulatee gene's expression. In this case, a regulatee gene has binding sites only for the TFs specific to its true regulators. Therefore, a gene can not be a true regulator if its TFs do not have binding sites at the regulatee gene. In case we possess the prior knowledge about a candidate regulator that its TFs do not have

binding sites at the regulatee, we can reliably reject the candidacy of that regulator. If such a candidate regulator's expression demonstrates a high correlation with that of the regulatee in the gene expression data, the proposed algorithms are likely to shortlist the candidate regulator. Such candidate regulators can be removed from the shortlist when TF-binding-site data are incorporated. Thus, false positive predictions can be reduced by extending the abilities of the proposed algorithms for incorporating complementary multi-omics data.

Data Integration

For a complex biological process, reconstruction of its time-varying gene regulatory networks enables us to understand the process at the gene-regulation level. However, such a process has multiple levels of biological regulations. For example, development of a cancer may involve mutations at the DNA-sequence level, consequent changes at the gene-regulation level, followed by alterations in the protein-expression level, ultimately resulting in uncontrolled cell growth and formations of tumours at the tissue level. Therefore, understanding the whole process requires reverse-engineering it from data acquired at different levels, such as - DNA sequence data, gene expression data, protein profiles and pathological test reports. Researchers are attempting to develop algorithms that can reconstruct a whole process by integrating different levels of biological data (Jain et al., 2016). Pursuing such developments shall be a worthwhile challenge; if successful, it will push the boundaries of our biological understanding and medical acumen.

Bibliography

Genexplain transfac®. URL http://genexplain.com/transfac/. As of Oct 10, 2017, the webpage claims that "TRANSFAC® is the database of eukaryotic transcription factors, their genomic binding sites and DNA-binding profiles. Dating back to a very early compilation, it has been carefully maintained and curated since then and became the gold standard in the field, which can be made use of when applying the geneXplain platform (http://genexplain.com/genexplain-platform).".

Amr Ahmed and Eric P Xing. Recovering time-varying networks of dependencies in social and biological studies. *Proceedings of the National Academy of Sciences*, 106 (29):11878–11883, 2009.

Sara Aibar, Carmen Bravo González-Blas, Thomas Moerman, Hana Imrichova, Gert Hulselmans, Florian Rambow, Jean-Christophe Marine, Pierre Geurts, Jan Aerts, Joost van den Oord, et al. Scenic: single-cell regulatory network inference and clustering. *Nature methods*, 14(11):1083–1086, 2017.

Uri Alon. *An introduction to systems biology: design principles of biological circuits*. CRC press, 2006.

Michelle N. Arbeitman, Eileen E. M. Furlong, Farhad Imam, Eric Johnson, Brian H. Null, Bruce S. Baker, Mark A. Krasnow, Matthew P. Scott, Ronald W. Davis, and Kevin P. White. Gene expression during the life cycle of drosophila melanogaster. *Science*, 297(5590):2270–2275, 2002. ISSN 0036-8075. doi: 10.1126/science.1072152. URL http://science.sciencemag.org/content/297/5590/2270.

ARTIVA. ARTIVA package. URL https://cran.r-project.org/package=ARTIVA. Last accessed: Dec 14, 2019.

Onureena Banerjee, Laurent El Ghaoui, and Alexandre d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *Journal of Machine Learning Research*, 9(Mar):485–516, 2008.

Mukesh Bansal, Vincenzo Belcastro, Alberto Ambesi-Impiombato, and Diego Di Bernardo. How to infer gene networks from expression profiles. *Molecular systems biology*, 3(1):78, 2007.

Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

Albert-Laszlo Barabasi and Zoltan N Oltvai. Network biology: understanding the cell's functional organization. *Nature reviews genetics*, 5(2):101–113, 2004.

Katia Basso, Adam A Margolin, Gustavo Stolovitzky, Ulf Klein, Riccardo Dalla-Favera, and Andrea Califano. Reverse engineering of regulatory networks in human b cells. *Nature genetics*, 37(4):382–390, 2005.

Nitin Bhardwaj, Matthew B. Carson, Alexej Abyzov, Koon-Kiu Yan, Hui Lu, and Mark B. Gerstein. Analysis of combinatorial regulation: Scaling of partnerships between regulators with the number of governed targets. *PLOS Computational Biology*, 6(5):1–9, 05 2010. doi: 10.1371/journal.pcbi.1000755. URL `https://doi.org/10.1371/journal.pcbi.1000755`.

David R Bickel. Probabilities of spurious connections in gene networks: application to expression time series. *Bioinformatics*, 21(7):1121–1128, 2005.

Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

Atul J Butte and Isaac S Kohane. Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements. In *Pacific Symposium on Biocomputing*, volume 5, pages 418–429, 2000.

India Central TB Division. TB India 2015, Revised National TB Control Programme, Annual Status Report, March 2015. URL `http://www.tbcindia.nic.in/WriteReadData/l892s/254998242TB%20India%202015.pdf`. Central TB Division, Directorate General of Health Services, Ministry of Health and Family Welfare, India.

Julien Chiquet, Yves Grandvalet, and Christophe Ambroise. Inferring multiple graphical structures. *Statistics and Computing*, 21(4):537–553, 2011.

Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012.

Patrick Danaher, Pei Wang, and Daniela M Witten. The joint graphical lasso for inverse covariance estimation across multiple classes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 76(2):373–397, 2014.

Alberto De La Fuente, Nan Bing, Ina Hoeschele, and Pedro Mendes. Discovery of meaningful associations in genomic data using partial correlation coefficients. *Bioinformatics*, 20(18):3565–3574, 2004.

D. Deriso. Is the inverse of a symmetric matrix also symmetric? `http://math.stackexchange.com/q/602192`, 2013. Author profile: `https://math.stackexchange.com/users/115025/d-deriso`. Last accessed: Dec 15, 2019.

Frank Dondelinger, Sophie Lèbre, and Dirk Husmeier. Non-homogeneous dynamic Bayesian networks with Bayesian regularization for inferring gene regulatory networks with gradually time-varying structure. *Machine Learning*, 90(2):191–230, 2013.

DREAM3. DREAM3 in silico network challenge. URL `https://www.synapse.org/#!Synapse:syn2853594/wiki/71567`. Last accessed: May 15, 2017.

EDISON. EDISON package. URL `https://cran.r-project.org/package=EDISON`. Last accessed: Dec 14, 2019.

David Edwards. *Introduction to graphical modelling*. Springer Science & Business Media, 2012.

Bradley Efron. Nonparametric estimates of standard error: the jackknife, the bootstrap and other methods. *Biometrika*, 68(3):589–599, 1981.

Michael B Eisen, Paul T Spellman, Patrick O Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

Jeremiah J Faith, Boris Hayete, Joshua T Thaden, Ilaria Mogno, Jamey Wierzbowski, Guillaume Cottarel, Simon Kasif, James J Collins, and Timothy S Gardner. Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles. *PLOS Biology*, 5(1):1–13, 01 2007. doi: 10.1371/journal.pbio.0050008. URL https://doi.org/10.1371/journal.pbio.0050008.

Alberto Franzin, Francesco Sambo, and Barbara Di Camillo. bnstruct: an R package for Bayesian Network structure learning in the presence of missing data. *Bioinformatics*, 33(8):1250–1252, 12 2016. ISSN 1367-4803. doi: 10.1093/bioinformatics/btw807. URL https://doi.org/10.1093/bioinformatics/btw807.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

Nir Friedman, Kevin Murphy, and Stuart Russell. Learning the structure of dynamic probabilistic networks. In *Proceedings of the Fourteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 139–147, San Francisco, CA, 1998. Morgan Kaufmann.

GENIE3. GENIE3 package. URL https://bioconductor.org/packages/GENIE3/. Last accessed: Apr 23, 2020.

GERBIL. Precision, Recall and F1 measure. URL https://github.com/dice-group/gerbil/wiki/Precision,-Recall-and-F1-measure. Section 'Dividing by 0'. Data Science Group at Paderborn University, Germany. Last accessed: Oct 13, 2017.

Gerstein Lab. Source Code of the B Team's Algorithm (BTA), 2010. URL http://info.gersteinlab.org/Dream3. Last accessed: Apr 23, 2020.

Biobase GmbH. Transfac public database version 7.0. URL http://gene-regulation.com/cgi-bin/pub/databases/transfac/search.cgi. The user requires to create a free-of-cost account to access the database. Last accessed: Oct 10, 2017.

Marco Grzegorczyk and Dirk Husmeier. Non-stationary continuous dynamic Bayesian networks. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 682–690. Curran Associates, Inc., 2009. URL http://papers.nips.cc/paper/3687-non-stationary-continuous-dynamic-bayesian-networks.pdf.

Marco Grzegorczyk and Dirk Husmeier. Improvements in the reconstruction of time-varying gene regulatory networks: dynamic programming and regularization by information sharing among genes. *Bioinformatics*, 27(5):693–699, 2011.

Vân Anh Huynh-Thu, Alexandre Irrthum, Louis Wehenkel, and Pierre Geurts. Inferring regulatory networks from expression data using tree-based methods. *PLOS ONE*, 5 (9):1–10, 09 2010. doi: 10.1371/journal.pone.0012776. URL https://doi.org/10.1371/journal.pone.0012776.

Niklas Jahnsson, Brandon Malone, and Petri Myllymäki. Duplicate detection for Bayesian network structure learning. *New Generation Computing*, 35(1):47–67, Jan 2017. doi: 10.1007/s00354-016-0004-9. URL https://doi.org/10.1007/s00354-016-0004-9.

Siddhartha Jain, Joel Arrais, Narasimhan J Venkatachari, Velpandi Ayyavoo, and Ziv Bar-Joseph. Reconstructing the temporal progression of HIV-1 immune response pathways. *Bioinformatics*, 32(12):i253–i261, 2016.

Yongsoo Kim, Seungmin Han, Seungjin Choi, and Daehee Hwang. Inference of dynamic networks using time-course data. *Briefings in bioinformatics*, 15(2):212–228, 2014.

Sophie Lèbre. *Stochastic process analysis for Genomics and Dynamic Bayesian Networks inference.* PhD thesis, Université d'Evry-Val d'Essonne, 2007.

Sophie Lèbre, Jennifer Becq, Frédéric Devaux, Michael PH Stumpf, and Gaëlle Lelandais. Statistical inference of the time-varying structure of gene-regulation networks. *BMC Systems Biology*, 4(1):130, Sep 2010. doi: 10.1186/1752-0509-4-130. URL `https://doi.org/10.1186/1752-0509-4-130`.

Fei Liu, Shao-Wu Zhang, Wei-Feng Guo, Ze-Gang Wei, and Luonan Chen. Inference of gene regulatory network based on local Bayesian networks. *PLOS Computational Biology*, 12(8):1–17, 08 2016. doi: 10.1371/journal.pcbi.1005024. URL `https://doi.org/10.1371/journal.pcbi.1005024`.

Zhi-Ping Liu. Quantifying gene regulatory relationships with association measures: A comparative study. *Frontiers in Genetics*, 8:96, 2017. ISSN 1664-8021. doi: 10.3389/fgene.2017.00096. URL `http://journal.frontiersin.org/article/10.3389/fgene.2017.00096`.

log0. Differences between L1 and L2 as loss function and regularization. `http://www.chioka.in/differences-between-l1-and-l2-as-loss-function-and-regularization/`, December 2013. Author profile: `https://github.com/log0`. Last accessed: Dec 15, 2019.

Paul M Magwene and Junhyong Kim. Estimating genomic coexpression networks using first-order conditional independence. *Genome biology*, 5(12):1, 2004.

Daniel Marbach, Thomas Schaffter, Claudio Mattiussi, and Dario Floreano. Generating Realistic In Silico Gene Networks for Performance Assessment of Reverse Engineering Methods. *Journal of Computational Biology*, 16(2):229–239, 2009. doi: 10.1089/cmb.2008.09TT. WingX.

Daniel Marbach, Robert J. Prill, Thomas Schaffter, Claudio Mattiussi, Dario Floreano, and Gustavo Stolovitzky. Revealing strengths and weaknesses of methods for gene network inference. *PNAS*, 107(14):6286–6291, 2010. doi: 10.1073/pnas.0913357107. WingX.

Daniel Marbach, Sushmita Roy, Ferhat Ay, Patrick E Meyer, Rogerio Candeias, Tamer Kahveci, Christopher A Bristow, and Manolis Kellis. Predictive regulatory models in drosophila melanogaster by integrative inference of transcriptional networks. *Genome research*, 22(7):1334–1349, 2012.

Adam A Margolin, Ilya Nemenman, Katia Basso, Chris Wiggins, Gustavo Stolovitzky, Riccardo D Favera, and Andrea Califano. ARACNE: an algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context. *BMC bioinformatics*, 7(Suppl 1):S7, 2006.

Florian Markowetz and Rainer Spang. Inferring cellular networks – a review. *BMC Bioinformatics*, 8(6):S5, Sep 2007. ISSN 1471-2105. doi: 10.1186/1471-2105-8-S6-S5. URL `https://doi.org/10.1186/1471-2105-8-S6-S5`.

Patrick E Meyer, Frederic Lafitte, and Gianluca Bontempi. minet: Ar/bioconductor package for inferring large transcriptional networks using mutual information. *BMC bioinformatics*, 9(1):461, 2008.

Karthik Mohan, Mike Chung, Seungyeop Han, Daniela Witten, Su-In Lee, and Maryam Fazel. Structured learning of Gaussian graphical models. In *Advances in neural information processing systems*, pages 620–628, 2012.

Kevin P. Murphy. The Bayes net toolbox for MATLAB. *Computing Science and Statistics*, 33:2001, 2001.

Kevin Patrick Murphy. How to use the Bayes net toolbox. URL `http://bayesnet.github.io/bnt/docs/usage.html`. Last accessed on Aug 15, 2017. The documentation was last updated on Oct 29, 2007.

Kevin Patrick Murphy. *Dynamic Bayesian networks: representation, inference and learning*. PhD thesis, University of California, Berkeley, 2002.

NACO and NIMS. India HIV Estimations, 2015. URL `http://naco.gov.in/upload/2015%20MSLNS/HSS/India%20HIV%20Estimations%202015.pdf`. National AIDS Control Organisation and National Institute of Medical Statistics, ICMR; Ministry of Health and Family Welfare, Government of India.

Manjari Narayan. What is the difference between L1 and L2 regularization? `https://www.quora.com/What-is-the-difference-between-L1-and-L2-regularization`, November 2014. Author profile: `https://www.quora.com/profile/Manjari-Narayan`. Last accessed: Dec 15, 2019.

Andrew Y Ng. Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, page 78. ACM, 2004.

India NICPR. National Institute of Cancer Prevention and Research (NICPR) statistics, July 2016. URL `http://cancerindia.org.in/cp/index.php/know-about-cancer/statistics`.

Olga Nikolova, Jaroslaw Zola, and Srinivas Aluru. Parallel globally optimal structure learning of Bayesian networks. *Journal of Parallel and Distributed Computing*, 73(8): 1039–1048, 2013.

Chris J Oates and Sach Mukherjee. Joint structure learning of multiple non-exchangeable networks. In *AISTATS*, pages 687–695, 2014.

Chris J Oates, Jim Korkola, Joe W Gray, Sach Mukherjee, et al. Joint estimation of multiple related biological networks. *The Annals of Applied Statistics*, 8(3):1892–1919, 2014.

Chris J Oates, Jim Q Smith, Sach Mukherjee, and James Cussens. Exact estimation of multiple directed acyclic graphs. *Statistics and Computing*, pages 1–15, 2015.

Judea Pearl. Bayesian networks: A model of self-activated memory for evidential reasoning. 1985.

Judea Pearl. Causality: models, reasoning and inference. *Econometric Theory*, 19: 675–685, 2003.

Roger Penrose. A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, volume 51, pages 406–413. Cambridge Univ Press, 1955.

Robert J. Prill, Daniel Marbach, Julio Saez-Rodriguez, Peter K. Sorger, Leonidas G. Alexopoulos, Xiaowei Xue, Neil D. Clarke, Gregoire Altan-Bonnet, and Gustavo Stolovitzky. Towards a rigorous assessment of systems biology models: The DREAM3 challenges. *PLOS ONE*, 5(2):1–18, 02 2010. doi: 10.1371/journal.pone.0009202. URL `https://doi.org/10.1371/journal.pone.0009202`.

Saptarshi Pyne and Ashish Anand. Rapid reconstruction of time-varying gene regulatory networks with limited main memory. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, pages 1–1, 2019a. doi: 10.1109/TCBB.2019.2946826. Early access.

Saptarshi Pyne and Ashish Anand. Capturing transient edges in time-varying gene regulatory networks. 2019b. In preparation.

Saptarshi Pyne, Alok Ranjan Kumar, and Ashish Anand. Rapid reconstruction of time-varying gene regulatory networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 17(1):278–291, Jan-Feb 2020. doi: 10.1109/TCBB.2018.2861698.

R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. URL `http://www.R-project.org`. ISBN 3-900051-07-0.

Alpan Raval and Animesh Ray. *Introduction to biological networks*. Chapman and Hall/CRC, 2016.

Joshua W Robinson and Alexander J Hartemink. Non-stationary dynamic Bayesian networks. In *Advances in neural information processing systems*, pages 1369–1376, 2009.

Guido Sanguinetti and Vân Anh Huynh-Thu, editors. *Gene Regulatory Networks: Methods and Protocols*, volume 1883 of *Methods in Molecular Biology book series*. Humana Press, 2019. URL `https://doi.org/10.1007/978-1-4939-8882-2`.

Juliane Schäfer and Korbinian Strimmer. An empirical Bayes approach to inferring large-scale gene association networks. *Bioinformatics*, 21(6):754–764, 2005.

Tomi Silander. Hyperparameter sensitivity revisited. In *Advanced Methodologies for Bayesian Networks*, pages 7–7, 2017.

Tomi Silander and Petri Myllymäki. A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, UAI'06, pages 445–452, Arlington, Virginia, United States, 2006. AUAI Press. ISBN 0-9749039-2-2. URL `http://dl.acm.org/citation.cfm?id=3020419.3020473`.

Abraham Silberschatz, Peter Baer Galvin, and James L Peterson. *Operating system concepts*. Addison-Wesley,, 1991.

Le Song, Mladen Kolar, and Eric P Xing. Keller: estimating time-varying interactions between genes. *Bioinformatics*, 25(12):i128–i136, 2009a.

Le Song, Mladen Kolar, and Eric P Xing. Time-varying dynamic Bayesian networks. In *Advances in Neural Information Processing Systems*, pages 1732–1740, 2009b.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Anthony Trewavas. A brief history of systems biology "Every object that biology studies is a system of systems." Francois Jacob (1974). *The Plant Cell*, 18(10):2420–2430, 2006.

Nathalie Villa-Vialaneix. Joint network inference with the consensual LASSO, April 2014. URL `http://www.nathalievilla.org/doc/pdf/slides_villavialaneix_etal_ENBISSM2014.pdf`. http://www.nathalievilla.org/.

Nathalie Villa-Vialaneix, Matthieu Vignes, Nathalie Viguerie, and Magali San Cristobal. Inferring networks from multiple samples with consensus lasso. *Quality Technology & Quantitative Management*, 11(1):39–60, 2014.

Ting Wang, Zhao Ren, Ying Ding, Zhou Fang, Zhe Sun, Matthew L MacDonald, Robert A Sweet, Jieru Wang, and Wei Chen. FastGGM: An efficient algorithm for the inference of Gaussian graphical model in biological networks. *PLoS Comput Biol*, 12(2):e1004755, 2016.

Wikipedia. Wikipedia entry on Cell (biology), a. URL `https://en.wikipedia.org/wiki/Cell_(biology)`.

Wikipedia. Wikipedia entry on Pseudolikelihood, b. URL `https://en.wikipedia.org/wiki/Pseudolikelihood`.

Wikipedia. Wikipedia entry on Quadratic programming, c. URL `https://en.wikipedia.org/wiki/Quadratic_programming#Problem_formulation`.

Anja Wille and Peter Bühlmann. Low-order conditional independence graphs for inferring genetic networks. *Statistical applications in genetics and molecular biology*, 5(1), 2006.

Anja Wille, Philip Zimmermann, Eva Vranová, Andreas Fürholz, Oliver Laule, Stefan Bleuler, Lars Hennig, Amela Prelić, Peter von Rohr, Lothar Thiele, et al. Sparse graphical Gaussian modeling of the isoprenoid gene network in arabidopsis thaliana. *Genome biology*, 5(11):1, 2004.

Eric P Xing, Wenjie Fu, Le Song, et al. A state-space mixed membership blockmodel for dynamic network tomography. *The Annals of Applied Statistics*, 4(2):535–566, 2010.

Kevin Y Yip, Roger P Alexander, Koon-Kiu Yan, and Mark Gerstein. Improved reconstruction of in silico gene regulatory networks by integrating knockout and perturbation data. *PloS one*, 5(1):e8121, 2010.

Xiangtian Yu, Guojun Li, and Luonan Chen. Prediction and early diagnosis of complex diseases by edge-network. *Bioinformatics*, 30(6):852–859, 10 2013. ISSN 1367-4803. doi: 10.1093/bioinformatics/btt620. URL `https://doi.org/10.1093/bioinformatics/btt620`.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Aimee K Zaas, Minhua Chen, Jay Varkey, Timothy Veldman, Alfred O Hero III, Joseph Lucas, Yongsheng Huang, Ronald Turner, Anthony Gilbert, Robert Lambkin-Williams, et al. Gene expression signatures diagnose influenza and other symptomatic respiratory viral infections in humans. *Cell host & microbe*, 6(3):207–217, 2009.

Publications

Manuscripts (Published)

- **Saptarshi Pyne** and Ashish Anand. Rapid Reconstruction of Time-varying Gene Regulatory Networks with Limited Main Memory. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*. Early access, 2019.
  doi: 10.1109/TCBB.2019.2946826 .
- **Saptarshi Pyne**, Alok Ranjan Kumar, and Ashish Anand. Rapid reconstruction of time-varying gene regulatory networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 17(1):278-291, Jan-Feb 2020.
  doi: 10.1109/TCBB.2018.2861698 .

Manuscripts (In Preparation)

- **Saptarshi Pyne** and Ashish Anand. Capturing Transient Edges in Time-varying Gene Regulatory Networks.

Software Developed

- **Saptarshi Pyne**, Manan Gupta and Ashish Anand. `R` package 'TGS'. Accepted and published at *Comprehensive R Archive Network* also known as *CRAN*. Online, URL: `https://cran.r-project.org/package=TGS` , 2018.

Appendix A

Pseudocodes

Algorithm 8 *TBN*

procedure TBN($\mathcal{D}$)
  Initialize $\mathcal{G} \leftarrow$ a null graph over $(V \times T)$ nodes.
  for each time interval $\left(t_p, t_{(p+1)}\right)$ **do**
    ▷ where $1 \leq p \leq (T-1)$; $(T-1)$ iterations.
    for each gene $v_j \in \mathcal{V}$ **do** ▷ (V) iterations.
      Candidate regulators of $v_j\_t_{(p+1)}$
        $\leftarrow \{v_i\_t_p : v_i \in \mathcal{V}\}$.
      Candidate regulator sets of $v_j\_t_{(p+1)}$
        $\leftarrow$ Powerset $(\{v_i\_t_p : v_i \in \mathcal{V}\})$.
      Find out a regulator set with the maximum
      BIC score by computing the scores of all
      candidate regulator sets from
      $\mathcal{D}_{\left(\mathcal{V};\{t_p,t_{(p+1)}\};\mathcal{S}\right)}$ using the *Bene* algorithm.
      Once the regulator set is finalised, for each
      node in it, add an edge in $\mathcal{G}$
      from that node to $v_j\_t_{(p+1)}$.
      ▷ $o\left(V^2 2^{(V-2)}\right)$ Silander and Myllymäki (2006)
    end for
  end for
  return $\mathcal{G}$.
end procedure

Algorithm 9 *CLR* Faith et al. (2007)

procedure CLR($\mathcal{D}, \mathscr{M}$)
    ($\mathscr{M}$ is the Mutual Information (MI) matrix. It is a
    $(V \times V)$ matrix. The $(v_i, v_j)^{th}$ cell of $\mathscr{M}$, denoted by
    $\mathscr{M}(v_i, v_j)$, represents the estimated MI value
    between v_i and v_j.)
    Initialize CLR network $\mathcal{G}_{\text{CLR}} \leftarrow$ a null graph over the
    genes in $\mathcal{V}$. ▷ $\mathcal{O}\left(V^2\right)$.
    for each pair of genes $\{v_i, v_j\}$ **do** ▷ $\mathcal{O}\left(V^2\right)$.
        Calculate CLR weight $w_{i,j} = \sqrt{z_i^2 + z_j^2}$ where
        $z_i = \max\left(0, \frac{\mathscr{M}(v_i,v_j) - \bar{v}_i}{\sigma(v_i)}\right)$ where, in turn,
        $(\bar{v}_i, \sigma(v_i))$ are the parameters of the empirical
        distribution, estimated from the MI values
        $\{\mathscr{M}(v_i, v_k) : v_k \in \mathcal{V} \setminus \{v_i\}\}$.
        Similarly, z_j is calculated.
        if $w_{i,j} > 0$ **then**
            Add an undirected edge in $\mathcal{G}_{\text{CLR}}$ between v_i
            and v_j with edge weight $w_{i,j}$.
        end if
    end for
    return $\mathcal{G}_{\text{CLR}}$.
end procedure

Algorithm 10 *TGS*

procedure TGS($\mathcal{D}$)
    Compute the Mutual Information (MI) matrix,
    denoted by $\mathscr{M}$. It is a $(V \times V)$ matrix. The $(v_i, v_j)^{th}$
    cell of $\mathscr{M}$, denoted by $\mathscr{M}(v_i, v_j)$, represents the
    estimated MI value between v_i and v_j. ▷ $\mathcal{O}\left(V^2\right)$.
    Initialize $\mathcal{G} \leftarrow$ a null graph over $(V \times T)$ nodes.
    $\mathcal{G}_{\text{CLR}} \leftarrow \text{CLR}(\mathcal{D}, \mathscr{M})$. ▷ (Algorithm 9), $\mathcal{O}\left(V^2\right)$.
    for each time interval $\left(t_p, t_{(p+1)}\right)$ **do**
        (where $1 \leq p \leq (T-1)$) ▷ $(T-1)$ iterations
        for each gene $v_j \in \mathcal{V}$ **do** ▷ (V) iterations
            Candidate regulators of $v_j\_t_{(p+1)} \leftarrow$
            $\{v_i\_t_p : (v_i, v_j) \in \text{Edgeset}(\mathcal{G}_{\text{CLR}})\}$.
            Candidate regulator sets of $v_j\_t_{(p+1)} \leftarrow$
            $\text{Powerset}(\{v_i\_t_p : (v_i, v_j) \in \text{Edgeset}(\mathcal{G}_{\text{CLR}})\})$.
            Find out a regulator set with the maximum
            BIC score by computing the scores of all
            candidate regulator sets from
            $\mathcal{D}_{(\mathcal{V};\{t_p, t_{(p+1)}\};\mathcal{S})}$ using the *Bene* algorithm.
            Once the regulator set is finalised, for each
            node in it, add an edge in $\mathcal{G}$ from
            that node to $v_j\_t_{(p+1)}$. Suppose,
            M = maximum number of neighbours any
            gene has in $\mathcal{G}_{\text{CLR}}$ Silander and Myllymäki (2006).
            ▷ $o\left(M^2 2^{(M-2)}\right)$.
        end for
    end for
    return $\mathcal{G}$.
end procedure

Algorithm 11 *TGS* with the Max Fan-in Restriction

procedure TGS($\mathcal{D}, M_f$)
  Compute the Mutual Information (MI) matrix,
  denoted by $\mathscr{M}$. It is a $(V \times V)$ matrix. The $(v_i, v_j)^{th}$
  cell of $\mathscr{M}$, denoted by $\mathscr{M}(v_i, v_j)$, represents the
  estimated MI value between v_i and v_j. ▷ $\mathcal{O}\left(V^2\right)$.
  Initialize $\mathcal{G} \leftarrow$ a null graph over $(V \times T)$ nodes.
  $\mathcal{G}_{\text{CLR}} \leftarrow \text{CLR}\,(\mathcal{D}, \mathscr{M})$. ▷ (Algorithm 9), $\mathcal{O}\left(V^2\right)$.
  for each time interval $\left(t_p, t_{(p+1)}\right)$ **do**
    (where $1 \leq p \leq (T-1)$) ▷ $(T-1)$ iterations
    for each gene $v_j \in \mathcal{V}$ **do** ▷ (V) iterations
      if $|\{v_i\_t_p\}| > M_f$ **then**
        (where $(v_i, v_j) \in \text{Edgeset}\,(\mathcal{G}_{\text{CLR}})$)
        Sort such v_i genes in descending order of
        the edge weight $w_{i,j}$ in $\mathcal{G}_{\text{CLR}}$. Generate a
        list L_j by retaining the top M_f number of
        genes and discarding the rest. Break ties
        using lexicographic order of the gene
        names or indices.
        Candidate regulators of $v_j\_t_{(p+1)} \leftarrow$
          $\{v_i\_t_p :$
          $((v_i, v_j) \in \text{Edgeset}\,(\mathcal{G}_{\text{CLR}})) \wedge (v_i \in L_j)\}$.
      else
        Candidate regulators of $v_j\_t_{(p+1)} \leftarrow$
          $\{v_i\_t_p : (v_i, v_j) \in \text{Edgeset}\,(\mathcal{G}_{\text{CLR}})\}$.
      end if
      Candidate regulator sets of $v_j\_t_{(p+1)} \leftarrow$
        Powerset$(\{v_i_t_p :$
        $(v_i, v_j) \in \text{Edgeset}\,(\mathcal{G}_{\text{CLR}})\})$.
      Find out a regulator set with the maximum
      BIC score by computing the scores of all
      candidate regulator sets from
      $\mathcal{D}_{\left(\mathcal{V};\{t_p, t_{(p+1)}\};\mathcal{S}\right)}$ using the *Bene* algorithm.
      Once the regulator set is finalised, for each
      node in it, add an edge in $\mathcal{G}$ from
      that node to $v_j\_t_{(p+1)}$. ▷ $o\left(M_f^2\ 2^{(M_f-2)}\right)$.
    end for
  end for
  return $\mathcal{G}$.
end procedure

Algorithm 12 *ARACNE* (Margolin et al., 2006)

procedure ARACNE($\mathscr{M}$)
    ▷ $\mathscr{M}$ is the raw Mutual Information (MI) matrix.
    $\mathscr{T} \leftarrow$ Initialize a $(V \times V)$ matrix.
        Assign zero to each cell. ▷ $\mathcal{O}\left(V^2\right)$.
    for each 3-combination of genes $\{v_i, v_j, v_k\}$ **do**
        ▷ $\mathcal{O}\left(\binom{V}{3}\right) = \mathcal{O}\left(V^3\right)$.
        if there exists a pair among
        $\{\{v_i, v_j\}, \{v_j, v_k\}, \{v_k, v_i\}\}$, whose MI value
        is less than the other two pairs **then**
            Tag that pair i.e.
            make $\mathscr{T}\left(v_i, v_j\right) \leftarrow 1$, assuming
            that (v_i, v_j) is such a pair.
        end if
    end for
    for each tagged pair (v_i, v_j) in $\mathscr{T}$ **do** ▷ $\mathcal{O}\left(V^2\right)$.
        Reset their raw MI value i.e.
        make $\mathscr{M}(v_i, v_j) \leftarrow 0$.
    end for
    return the refined mutual information matrix $\mathscr{M}$.
end procedure

Algorithm 13 *TGS+* with the Max Fan-in Restriction

procedure TGS+$(\mathcal{D}, M_f)$
  Compute the Mutual Information (MI) matrix,
  denoted by $\mathscr{M}$. It is a $(V \times V)$ matrix. The $(v_i, v_j)^{th}$
  cell of $\mathscr{M}$, denoted by $\mathscr{M}(v_i, v_j)$, represents the
  estimated MI value between v_i and v_j. ▷ $\mathcal{O}\left(V^2\right)$.
  Refine $\mathscr{M}$ by passing it through *ARACNE* i.e.
  $\mathscr{M} \leftarrow$ ARACNE$(\mathscr{M})$. ▷ (Algorithm 12), $\mathcal{O}\left(V^3\right)$.
  Initialize $\mathcal{G} \leftarrow$ a null graph over $(V \times T)$ nodes.
  $\mathcal{G}_{\text{CLR}} \leftarrow \text{CLR}(\mathcal{D}, \mathscr{M})$. ▷ (Algorithm 9), $\mathcal{O}\left(V^2\right)$.
  for each time interval $\left(t_p, t_{(p+1)}\right)$ **do**
    (where $1 \leq p \leq (T-1)$) ▷ $(T-1)$ iterations
    for each gene $v_j \in \mathcal{V}$ **do** ▷ (V) iterations
      if $|\{v_i\_t_p\}| > M_f$ **then**
        (where $(v_i, v_j) \in \text{Edgeset}(\mathcal{G}_{\text{CLR}})$)
        Sort such v_i genes in descending order of
        the edge weight $w_{i,j}$ in $\mathcal{G}_{\text{CLR}}$. Generate a
        list L_j by retaining the top M_f number of
        genes and discarding the rest. Break ties
        using lexicographic order of the gene
        names or indices.
        Candidate regulators of $v_j\_t_{(p+1)} \leftarrow$
          $\{v_i\_t_p :$
          $((v_i, v_j) \in \text{Edgeset}(\mathcal{G}_{\text{CLR}})) \wedge (v_i \in L_j)\}$.
      else
        Candidate regulators of $v_j\_t_{(p+1)} \leftarrow$
          $\{v_i\_t_p : (v_i, v_j) \in \text{Edgeset}(\mathcal{G}_{\text{CLR}})\}$.
      end if
      Candidate regulator sets of $v_j\_t_{(p+1)} \leftarrow$
        Powerset$(\{v_i_t_p :$
        $(v_i, v_j) \in \text{Edgeset}(\mathcal{G}_{\text{CLR}})\})$.
      Find out a regulator set with the maximum
      BIC score by computing the scores of all
      candidate regulator sets from
      $\mathcal{D}_{(\mathcal{V};\{t_p, t_{(p+1)}\};\mathcal{S})}$ using the *Bene* algorithm.
      Once the regulator set is finalised, for each
      node in it, add an edge in $\mathcal{G}$ from
      that node to $v_j\_t_{(p+1)}$. ▷ $o\left(M_f^2\ 2^{(M_f-2)}\right)$.
    end for
  end for
  return $\mathcal{G}$.
end procedure

Algorithm 14 *Gen-next-set*

```
1: procedure GEN-NEXT-SET(curr.set)
2:     ## curr.set: current candidate regulator set.
3:     ******************************
4:     l ← |curr.set|.                          ▷ Θ(|curr.set|) = O(M_f).
5:     Initialize the carry bit with TRUE i.e.
6:     carry.bit ← 1_b.
7:     ## 1_b : Boolean TRUE, 0_b : Boolean FALSE.
8:     ******************************
9:     for bit index i = l to 1 do              ▷ Θ(l) = O(M_f).
10:        curr.bit ← curr.set[i] where
11:        curr.set[i] denotes the i-th bit of curr.set.
12:        ## curr.set[l] = the least significant bit of curr.set.
13:        ## curr.set[1] = the most significant bit of curr.set.
14:        if curr.bit and carry.bit both are 0_b then
15:            curr.set[i] ← 0_b.
16:            carry.bit ← 0_b.
17:        else if curr.bit and carry.bit both are 1_b then
18:            curr.set[i] ← 0_b.
19:            carry.bit ← 1_b.
20:        else
21:            curr.set[i] ← 1_b.
22:            carry.bit ← 0_b.
23:        end if
24:    end for
25:    ******************************
26:    return curr.set.
27:    ## It is an in-place replacement i.e. the input value
28:    ## of curr.set is modified in-place to generate the
29:    ## next value of curr.set, without creating any
30:    ## additional variable.
31: end procedure
```

Algorithm 15 *Find-best-set-Lite*

procedure FIND-BEST-SET-LITE($v_j\_t_{(p+1)}, \mathcal{V}_{(j;(p+1))}, \mathcal{D}^*$)
    ## $v_j\_t_{(p+1)}$: regulatee gene;
    ## $\mathcal{V}_{(j;(p+1))}$: candidate regulators of $v_j\_t_{(p+1)}$;
    ## $\mathcal{D}^*$: data needed to calculate the required BIC
    ## scores; in this particular case,
    ## $\mathcal{D}^* = \mathcal{D}_{\left(\{v_j\_t_{(p+1)}\} \cup \mathcal{V}_{(j;(p+1))}; \{t_p, t_{(p+1)}\}; \mathcal{S}\right)}$.
    ****
    curr.set ← empty set.
    best.set ← curr.set.
    best.score ← BIC score of curr.set.
        ▷ (Section 4.2, supplementary document, (Pyne and Anand, 2019a)),
        ▷ $T_{\text{BIC}}\left(V; T; S; M_f; \vec{\delta}\right)$.
    for loop counter = 2 to $2^{|\mathcal{V}_{(j;(p+1))}|}$ **do**
        ▷ $\Theta\left(2^{|\mathcal{V}_{(j;(p+1))}|} - 1\right) = \mathcal{O}\left(2^{M_f} - 1\right)$
        ▷ iterations.
        curr.set ← GEN-NEXT-SET(curr.set).
            ▷ Algorithm 14 , $\mathcal{O}\left(M_f\right)$.
        curr.score ← BIC score of curr.set.
            ▷ $T_{\text{BIC}}\left(V; T; S; M_f; \vec{\delta}\right)$.
        if curr.score > best.score **then**
            best.score ← curr.score.
            best.set ← curr.set.
        end if
    end for
    ****
    return best.set.
end procedure

Algorithm 16 *TGS-Lite* with the Max Fan-in Restriction

procedure TGS-LITE($\mathcal{D}, M_f$)
    ## $\mathcal{D}$: data; M_f : max fan-in.
    Compute the Mutual Information (MI) matrix,
    denoted by $\mathscr{M}$. It is a $(V \times V)$ matrix. The $(v_i, v_j)^{th}$
    cell of $\mathscr{M}$, denoted by $\mathscr{M}(v_i, v_j)$, represents the
    estimated MI value between v_i and v_j. ▷ $\mathcal{O}\left(V^2\right)$.
    Initialize $\mathcal{G} \leftarrow$ a null graph over $(V \times T)$ nodes.
    $\mathcal{G}_{\text{CLR}} \leftarrow \text{CLR}\left(\mathcal{D}, \mathscr{M}\right)$.
    ▷ (Algorithm 9), $\mathcal{O}\left(V^2\right)$.
    ****
    for each gene $v_j \in \mathcal{V}$ **do** ▷ (V) iterations
        for each time interval $\left(t_p, t_{(p+1)}\right)$ **do**
            (where $1 \leq p \leq (T-1)$) ▷ $(T-1)$ iterations
            ****
            if No. of neighbours of v_j in $\mathcal{G}_{\text{CLR}} > M_f$ **then**
                From the set of neighbours of v_j in $\mathcal{G}_{\text{CLR}}$,
                generate a list L_j by selecting the top
                M_f number of neighbours w.r.t. their
                edge weights with v_j in $\mathcal{G}_{\text{CLR}}$. Break
                ties using the lexicographic order of
                gene names or indices.
                $\mathcal{V}_{(j;(p+1))} \leftarrow$
                    $\{v_i\_t_p :$
                    $((v_i, v_j) \in \text{Edgeset}\left(\mathcal{G}_{\text{CLR}}\right)) \wedge (v_i \in L_j)\}$
                    where $\mathcal{V}_{(j;(p+1))}$: The set of
                    candidate regulators of $v_j\_t_{(p+1)}$.
                ▷ $\mathcal{O}\left(V\right)$.
            else
                $\mathcal{V}_{(j;(p+1))} \leftarrow$
                    $\{v_i\_t_p : (v_i, v_j) \in \text{Edgeset}\left(\mathcal{G}_{\text{CLR}}\right)\}$.
                ▷ $\mathcal{O}\left(M_f\right)$.
            end if
            ****
            best.set $\leftarrow$
            FIND-BEST-SET-LITE$\Big(v_j_t_{(p+1)}, \mathcal{V}_{(j;(p+1))},$
                $\mathcal{D}_{\left(\{v_j\_t_{(p+1)}\} \cup \mathcal{V}_{(j;(p+1))}; \{t_p, t_{(p+1)}\}; \mathcal{S}\right)}\Big)$.
                ▷ Algorithm 15 ,
                ▷ $T_{\text{Find-best-set-Lite}}\left(V; T; S; M_f; \vec{\delta}\right)$.
            for each node in best.set **do**
                ▷ $\Theta\left(|\text{best.set}|\right) = \mathcal{O}\left(M_f\right)$.
                Add an edge in $\mathcal{G}$
                from that node to $v_j\_t_{(p+1)}$.
            end for
        end for
    end for
    ****
    return $\mathcal{G}$.
end procedure

Algorithm 17 *TGS-Lite+* with the Max Fan-in Restriction

procedure TGS-LITE+$(\mathcal{D}, M_f)$
  ## $\mathcal{D}$: data; M_f : max fan-in.
  Compute the Mutual Information (MI) matrix,
  denoted by $\mathscr{M}$. It is a $(V \times V)$ matrix. The $(v_i, v_j)^{th}$
  cell of $\mathscr{M}$, denoted by $\mathscr{M}(v_i, v_j)$, represents the
  estimated MI value between v_i and v_j. ▷ $\mathcal{O}\left(V^2\right)$.
  Refine $\mathscr{M}$ by passing it through *ARACNE* i.e.
  $\mathscr{M} \leftarrow$ ARACNE$(\mathscr{M})$.
  ▷ (Algorithm 12), $\mathcal{O}\left(V^3\right)$.
  Initialize $\mathcal{G} \leftarrow$ a null graph over $(V \times T)$ nodes.
  $\mathcal{G}_{\text{CLR}} \leftarrow \text{CLR}\left(\mathcal{D}, \mathscr{M}\right)$.
  ▷ (Algorithm 9), $\mathcal{O}\left(V^2\right)$.
  for each gene $v_j \in \mathcal{V}$ **do** ▷ (V) iterations
    for each time interval $\left(t_p, t_{(p+1)}\right)$ **do**
      (where $1 \leq p \leq (T-1)$) ▷ $(T-1)$ iterations
      if No. of neighbours of v_j in $\mathcal{G}_{\text{CLR}} > M_f$ **then**
        From the set of neighbours of v_j in $\mathcal{G}_{\text{CLR}}$,
        generate a list L_j by selecting the top
        M_f number of neighbours w.r.t. their
        edge weights with v_j in $\mathcal{G}_{\text{CLR}}$. Break
        ties using the lexicographic order of
        gene names or indices.
        $\mathcal{V}_{(j;(p+1))} \leftarrow$
          $\{v_i\_t_p :$
          $((v_i, v_j) \in \text{Edgeset}\left(\mathcal{G}_{\text{CLR}}\right)) \wedge (v_i \in L_j)\}$
          where $\mathcal{V}_{(j;(p+1))}$: The set of
          candidate regulators of $v_j\_t_{(p+1)}$.
        ▷ $\mathcal{O}\left(V\right)$.
      else
        $\mathcal{V}_{(j;(p+1))} \leftarrow$
          $\{v_i\_t_p : (v_i, v_j) \in \text{Edgeset}\left(\mathcal{G}_{\text{CLR}}\right)\}$.
        ▷ $\mathcal{O}\left(M_f\right)$.
      end if
      best.set $\leftarrow$
      FIND-BEST-SET-LITE$\Big(v_j_t_{(p+1)}, \mathcal{V}_{(j;(p+1))},$
        $\mathcal{D}_{\left(\{v_j\_t_{(p+1)}\} \cup \mathcal{V}_{(j;(p+1))}; \{t_p, t_{(p+1)}\}; \mathcal{S}\right)}\Big)$.
      ▷ Algorithm 15 ,
      ▷ $T_{\text{Find-best-set-Lite}}\left(V; T; S; M_f; \vec{\delta}\right)$.
      for each node in best.set **do**
        ▷ $\Theta\left(|\text{best.set}|\right) = \mathcal{O}\left(M_f\right)$.
        Add an edge in $\mathcal{G}$
        from that node to $v_j\_t_{(p+1)}$.
      end for
    end for
  end for
  return $\mathcal{G}$.
end procedure