Rapid Reconstruction of Time-varying Gene Regulatory Networks

Saptarshi Pyne, Alok Ranjan Kumar, and Ashish Anand

Abstract—Rapid advancement in high-throughput gene expression measurement technologies has resulted in genome-scale time series datasets. Uncovering the underlying temporal sequence of gene regulatory events in the form of time-varying Gene Regulatory Networks (GRNs) demands computationally fast, accurate and highly scalable algorithms. An existing algorithm, namely *ARTIVA*, provides a flexible framework. It learns the time-varying GRN structures independently of each other, without imposing any structural constraint. However, it is time intensive and hence not suitable for large-scale GRNs. To provide a similarly flexible framework but in a significantly more time-efficient manner, a novel algorithm, namely *TGS*, is proposed here. *TGS* is shown to consume only 29 minutes for a microarray dataset with 4028 genes. Moreover, it provides the flexibility and time-efficiency, without losing the accuracy. It consistently outperforms ARTIVA in true positive detection on three realistically simulated benchmark datasets. Nevertheless, *TGS*'s main memory requirement grows exponentially with the number of genes, which it tackles by restricting the maximum number of regulators for each gene. Relaxing this restriction remains an important challenge as the true number of regulators is not known a prior. Links to all the datasets, source code and results of this paper are available in the supplementary document.

Index Terms—Gene Regulatory Network, Network Reconstruction, Bayesian Network, Probabilistic Graphical Model, Gene Expression, Temporal Progression Model, Network Inference, Structure Learning, Computational Systems Biology.

1 Introduction

Cell, the building block of life, is a dynamic system. It continuously senses and responds to the changes in its environmental conditions. The cell performs this task with the help of several biomolecules that interact with each other. Modelling how these interactions vary with time is critical for understanding how the cell develops, evolves and maintains itself. One of the important types of interactions is the ones between the Transcription Factors (TFs) and the genes. Each TF is a special type of protein which physically binds to an appropriate site in the vicinity of its target gene to regulate the gene's expression. The gene, that has produced the TF in the first place, is said to be a regulator of the target gene. This regulator-regulatee relationships among the genes is represented by a directed network, known as the Gene Regulatory Network (GRN). In a GRN, each node represents a gene and a directed edge from one gene to another implies that the former gene regulates the latter. The aim of this paper is to develop an algorithm for modelling (reconstructing) how the GRN structure (edge relationships) varies with time; in other words, which gene regulates which gene and when.

There exists an array of algorithms that reconstruct time-varying GRNs from time series gene expression datasets. Among them, the *ARTIVA* algorithm [1] demonstrates its ability to accomplish this task with high accuracy. It employs a flexible framework where the GRN structures are learnt independently of each other, without imposing any structural constraint. However, *ARTIVA* is very time intensive; therefore, not suitable for the contemporary large-scale (high-throughput) gene expression datasets.

In this paper, a novel algorithm, namely *TGS*, is proposed to provide an equally flexible framework in a significantly more time-efficient manner. It is shown that *TGS* requires only 29 minutes for a D. melanogaster life cycle microarray dataset with with 4028 genes.

It needs to be noted that, another existing algorithm, namely *TV-DBN* [2] is also able to demonstrate its scalability to a microarray dataset with 3626 genes (runtime is unknown). Nevertheless, TV-DBN accomplishes such scalability by imposing an explicit structural constraint, called smoothly time-varying assumption. It assumes that each GRN shares more common edges with its temporally adjacent GRNs than with the distal ones.

The smoothly time-varying assumption does not necessarily hold for all scenarios. For example, variation is highly dynamic during the Yeast stress response [1, Section 'Conclusions']. Another exception may arise when sampling interval of the given dataset is considerably large. In that case, two consecutive time points may belong to different cellular conditions. Thus, it is possible for a structure to share less common edges with a temporally adjacent structure belonging to a separate condition than with a distal structure belonging to a similar condition.

TGS's framework is compatible with any dataset regardless of whether the smoothly time-varying assumption holds for it or not. Moreover, *TGS* offers the flexibility and time-efficiency without losing the accuracy. DREAM3 in silico network inference challenge datasets are used to evaluate *TGS*'s accuracy in comparison to that of *ARTIVA*. The results show that *TGS* consistently outperforms *ARTIVA* in terms of true positive detection. Additionally, *TGS* is able to reconstruct biologically meaningful GRNs from a D. melanogaster microarray dataset.

To summarize, the main contribution of this paper is two-fold:

- **Flexibility:** It provides a framework where time-varying GRN structures are learnt independently of

each other, without imposing any structural constraint. This framework is compatible with any time series gene-expression dataset regardless of whether the underlying gene regulation process complies with the smoothly time-varying assumption or not.

- **Time-efficiency:** The framework is offered in a significantly more time-efficient manner than the state-of-the-art alternative; at the same time, consistently outperforming the alternative in terms of true positive detection.

This paper is organized as follows: Section 2 formally defines the problem statement at hand, and provides a critical and chronological survey of the related algorithms to identify the fundamental issues that need to be solved. Subsequently in Section 3, a novel algorithm is designed to resolve a selected subset of these issues. The learning power and speed of the algorithm are comparatively studied against those of the prior state-of-the-art algorithm on a set of benchmark synthetic datasets in Section 4. In the same section, biological significance of its prediction is evaluated, given a real dataset. Nevertheless, a set of limitations remain in its unconstrained application. These limitations are discussed in Section 5, which opens the door for future investigation.

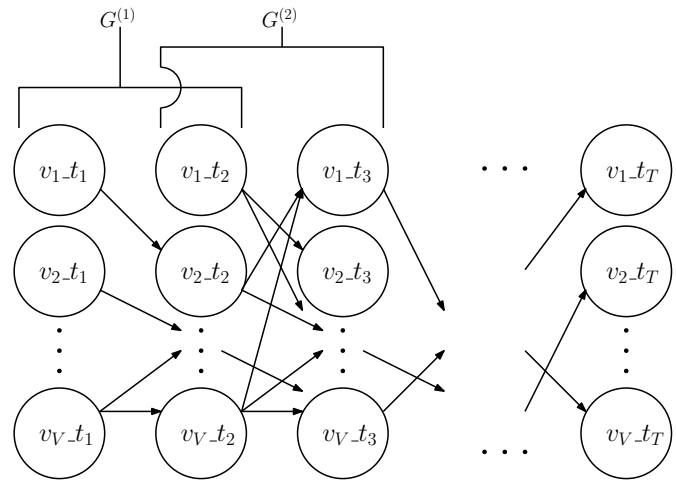


Fig. 1. Output time-varying GRNs $\left(G^{(1)}, \ldots, G^{(T-1)}\right) = \mathcal{G}$ is a sequence of directed unweighted networks. Here, $G^{(p)}\,(\in \mathcal{G})$ represents the gene regulatory events occurred during the time interval between time points t_p and $t_{(p+1)}$. It consists of $(2 \times V)$ 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 $\left(t_p, t_{(p+1)}\right)$.

2 Related Works

2.1 Problem Formulation

Suppose that the given dataset $\mathcal{D}$ is comprised of S time series $\mathcal{S} = \{s_1, \ldots, s_S\}$ of gene expression data. Each time series contains expression levels of V 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 is no missing value in any time series. In other words, each time series is a complete time series of T time points. $\mathcal{D}_{(\mathcal{X};\mathcal{Y};\mathcal{Z})}$ is used to denote the observed values of genes $\mathcal{X}$ at time points $\mathcal{Y}$ in time series $\mathcal{Z}$. $\mathcal{D}_{(\mathcal{X};\mathcal{Y};\mathcal{Z})} \subseteq \mathcal{D}$ since $\mathcal{X} \subseteq \mathcal{V}, \mathcal{Y} \subseteq \mathcal{T}, \mathcal{Z} \subseteq \mathcal{S}$.

Given $\mathcal{D}$, the objective is to reconstruct a temporally ordered GRN sequence $\mathcal{G} = \left(G^{(1)}, \ldots, G^{(T-1)}\right)$ (Figure 1). 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 $v_i_t_q$. The underlying gene regulation process is assumed to be first order Markovian [3, p. 140, Section 2]. Hence, there exists a directed edge $\left(v_i_t_p, v_j_t_{(p+1)}\right)$ if and only if v_i's expression at time point t_p has a regulatory effect on v_j's expression at time point $t_{(p+1)}$.

2.2 Existing Solutions

There exists an array of algorithms (e.g., *Bene* [4], *GENIE3* [5], *NARROMI* [6], *LBN* [7]) that do not completely solve the problem but can reconstruct a single 'summary' GRN over the nodes in $\mathcal{V}$. These algorithms consider $\mathcal{D}$ as a cross-sectional dataset. Then they include edge (v_i, v_j) in the summary GRN if and only if the expression level of v_j is not conditionally independent of that of v_i, given the expression levels of other genes. The summary GRN helps to discover which gene regulates which gene. However, it can not identify the time interval(s) during which such regulation has taken place.

To address this problem, Friedman et al. [3] model $\mathcal{G}$ as a Dynamic Probabilistic Network (or Dynamic Bayesian Network, in short, *DBN*). It is assumed that each gene is regulated by the same regulator(s), if any, at every time interval. Thus, output time-varying GRNs possess time-homogeneous structures [Section 1] [8] , e.g., edge $(v_i_t_1, v_j_t_2)$ exists in $G^{(1)}$ if and only if edge $(v_i_t_2, v_j_t_3)$ exists in $G^{(2)}$ and so on [3, Figure 1.(b)]. In practice, this approach tends to discover the regulations that are active over a large number of time intervals. However, they may miss the regulations that are active over a small number of time intervals [2, p. 2, Section 1].

To tackle the time-homogeneity issue, four algorithms are proposed. They are Non-stationary dynamic Bayesian networks (*NsDbn*) [9], Non-stationary continuous dynamic Bayesian networks (*NsCdbn*) [10], *ARTIVA* [1] and *cpBGe* [8]. These algorithms accommodate the case where the same gene can be regulated by different regulators during different time intervals. The class of resulting models is known as time-inhomogeneous DBN. *NsDbn* assumes that the data is generated by a multiple change point process with change points at $\mathfrak{T} = \{t_1, t_{(1+p)}, t_{(1+p+q)}, \ldots, t_T\} \subseteq \mathcal{T}$ where $p, q \in \mathbb{Z}_+$. The duration between two consecutive change points is called a time segment. One GRN structure is learnt for each time segment instead of each time interval. GRN structures of two different time segments can be different but GRN structure within a time segment remains unchanged. First, *NsDbn* determines the number of the change points along with their positions. Then for each time

segment, the fittest (w.r.t. a scoring function) DBN structure is identified to be the output GRN for that segment. The scoring function requires the input dataset to be discretized. In contrast, *NsCdbn* uses a different scoring function that does not require the dataset to be discretized [10, Section 1].

ARTIVA provides a more flexible model by relaxing the assumption that the change points are specific to all the genes. It assumes that each gene has its own set of change points (i.e. time segments specific to itself); therefore, it can be regulated by different regulators at its different time segments but within a time segment, it must be regulated by the same regulators. Hence, the output is one GRN structure for each time interval. If a particular time segment of a specific gene spans multiple consecutive time intervals, then the gene's regulator configurations (incoming edges) remain the same in the corresponding time interval specific GRN structures. On the other hand, if two consecutive time intervals belong to two separate time segments for a specific gene, then the gene's regulator configurations vary in the corresponding time interval specific GRN structures. Thus, *ARTIVA* learns the time interval specific GRN structures independently of each other [1, Section 'Conclusions'].

In the *cpBGe* paper, it is argued that the assumption of *NsDbn* is over-restrictive (same change points for all the genes) and that of *ARTIVA* is over-flexible (unique change points for every gene) [8, p. 694, Section 1]. Being true to the argument, *cpBGe* groups the genes into clusters based on similar expression patterns and then infers unique change points for every cluster. However, *cpBGe* produces a single time-invariant GRN structure. Therefore, *ARTIVA* remains the most viable alternative for reconstructing time-varying GRNs till that point.

However, *ARTIVA* is a computationally expensive algorithm. Its authors suggest that it requires approximately $(5 \times V)$ minutes to reconstruct GRNs from a dataset of $T = 20$ and $S = 5$ with the default parameter settings on a 2.66 GHz CPU having a 4 GB RAM [1, Section 'Technical information']. Hence, the time frame necessary for it to scale up to the high-throughput datasets may be considered prohibitive ($\simeq 87$ days for $V = 25,000$).

To address the scalability issue, the Time-Varying DBN (*TV-DBN*) [2] algorithm imposes an explicit structural constraint on the time-varying GRN structures. This constraint is called the smoothly time-varying assumption. It assumes that each GRN structure shares more common edges with its temporally adjacent GRN structures than with the distal ones [2, Section 4]. For example, $G^{(t_p)}$ is assumed to share more common edges with $G^{(t_p+1)}$ than with $G^{(t_p+2)}$. Hence in *TV-DBN*, the GRN structures in $\mathcal{G}$ are not learnt independently but they are learnt jointly. It formulates a highly-scalable penalized linear auto-regression based optimization problem [2, Eqn. (5)] that (a) regresses (predicts) expression level of each gene v_i at time point $t_p \in (\mathcal{T} \setminus t_1)$ with that of all genes $\mathcal{V}$ at time point $t_{(p-1)}$; then attempts to minimize the least square prediction error, and (b) penalizes the structural difference (uncommon edges) between two GRNs in proportion to their temporal distance to conform with the smoothly time-varying assumption. Its result is demonstrated for a $(V = 3626, T = 25, S = 2)$ dataset [2, Section 'Yeast Gene Regulatory Networks']).

The smoothly time-varying assumption is realistic when the system under consideration evolves smoothly with time. However, there are systems where the underlying gene regulation process is highly dynamic, e.g., the yeast stress response and the Drosophila development. In such scenarios, avoiding any assumption is more appropriate [1, Section 'Conclusions']. A detailed discussion on the limitations of the smoothly time-varying assumption can be found in Section 4.1 of the supplementary document.

Given a dataset, it is always preferable to avoid imposing any structural constraint, unless the constraint is known a prior to hold for the dataset. *ARTIVA* offers such an unconstrained framework. It learns each time interval specific GRN structure in a purely data-driven manner, independently of other GRN structures. However, *ARTIVA*'s high demand for computational time hinders its application with contemporary high-throughput datasets. Developing an unconstrained as well as time-efficient framework can be considered a timely contribution.

TABLE 1
A Summary of the existing GRN Reconstruction Algorithms, discussed in Section 2.2, in the context of Time Series Gene Expression Data.

Algorithm	Summary
Bene [4], *GENIE3* [5], *NARROMI* [6], *LBN* [7], *cpBGe* [8]	Learn a time-invariant summary GRN. Can not identify the time interval(s) during which a regulatory event has taken place.
DBN [3]	Learns time-varying GRNs. Identifies the time interval(s) during which a regulatory event has taken place. However, does not accommodate the case where a gene can be regulated by different regulators during different time intervals.
NsDbn [9], *NsCdbn* [10], *ARTIVA* [1]	Accommodate the case where a gene can be regulated by different regulators during different time intervals. Nevertheless, not suitable for large-scale datasets due to high computational cost.
TV-DBN [2]	Highly scalable. However, requires the assumption that each GRN shares more common edges with its temporally adjacent GRNs than the distal ones. It is called the smoothly time-varying assumption.

3 METHODS

In this section, two algorithms are developed. First, a baseline algorithm is developed for reconstructing time-varying GRNs from time series gene expression data. Like ARTIVA [1, 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 heuristic based approximation step is added to the baseline algorithm to develop the final algorithm. It maintains the independently time-varying framework without compromising the scalability.

3.1 Development of the Baseline Algorithm

A conditional independence based baseline algorithm, referred to as Time-varying Bayesian Networks (*TBN*), is designed. Algorithm 1 describes the steps in *TBN*. It takes a discretized 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 1). Each $G^{(p)}$ is modelled as a Bayesian Network (BN) [11, Section 'Bayesian networks']. Absence of a directed edge $\left(v_i_t_p, v_j_t_{(p+1)}\right)$ in $G^{(t_p)}$ 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 [12, 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 authors' knowledge. Hence, *TBN* with *Bene* is chosen as the baseline for developing a novel algorithm.

In *TBN* (Algorithm 1 Line 7), BIC scoring function [11, 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. [4, Section 4.4]. It is observed that BIC outperforms BDe when number of observations being considered (here, $(S+1)$) is below 20. Moreover, the performance of BDe is very sensitive to the chosen value of its hyper-parameter [13]. BIC, on the other hand, does not depend on any hyper-parameter. For these reasons, BIC is considered to be most suitable for the current study.

Algorithm 1 *TBN* algorithm

1: **procedure** TBN($\mathcal{D}$)
2: Initialize $\mathcal{G} \leftarrow$ a null graph over $(V \times T)$ nodes.
3: **for** each time interval $\left(t_p, t_{(p+1)}\right)$ where $1 \leq p \leq (T-1)$ **do** ▷ $(T-1)$ iterations
4: **for** each gene $v_j \in \mathcal{V}$ **do** ▷ (V) iterations
5: Candidate regulators of $v_j_t_{(p+1)} \leftarrow \{v_i_t_p : v_i \in \mathcal{V}\}$.
6: Candidate regulator sets of $v_j_t_{(p+1)} \leftarrow powerset\left(\{v_i_t_p : v_i \in \mathcal{V}\}\right)$.
7: 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 *Bene* algorithm. Once the regulator set is finalized, for each node in it, add an edge in $\mathcal{G}$ (Figure 1) from that node to $v_j_t_{(p+1)}$. ▷ $o\left(V^2 2^{(V-2)}\right)$ [4, Section 5]
8: **end for**
9: **end for**
10: **return** $\mathcal{G}$.
11: **end procedure**

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

From Algorithm 1 , it is found that *TBN*'s time complexity is $T_{\text{TBN}}(V) = (T-1) \times V \times o\left(V^2 2^{(V-2)}\right) = o\left((T-1)\, V^3 2^{(V-2)}\right)$. It 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 shortlisting 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 14 such measures are comparatively studied by Liu et al. [14] who conclude that Mutual Information (MI) demonstrates superior stability over other measures. MI's potential regulator-regulatee association predictions consistently outperform [14, 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 [7, 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 prior so that quality of a prediction can be measured. A more practical strategy is appointed by Context Likelihood of Relatedness (*CLR*) algorithm [15] (Algorithm 2). It constructs 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 inference algorithms

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 inference algorithm itself.

[15, 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 'the algorithm for reconstructing Time-varying Gene regulatory networks with Shortlisted candidate regulators'). A graphical flowchart is presented in Figure 2.

Algorithm 2 *CLR* algorithm

procedure CLR($\mathcal{D}$)
  Initialize CLR network $\mathcal{G}_{\text{CLR}} \leftarrow$ a null graph over the genes in $\mathcal{V}$.
  Compute MI between each pair of genes. $\triangleright\ \mathcal{O}\left(V^2\right)$.
  for each pair of genes $\{v_i, v_j\}$ **do** $\triangleright\ \mathcal{O}\left(V^2\right)$.
    Calculate CLR weight $w_{i,j} = \sqrt{z_i^2 + z_j^2}$ where
    $z_i = \max\left(0, \frac{\text{MI}(v_i,v_j)-\bar{v_i}}{\sigma(v_i)}\right)$ where in turn
    $(\bar{v_i}, \sigma(v_i))$ are the parameters of the empirical distribution, estimated from MI values $\{\text{MI}(v_i, v_k) : v_k \in \mathcal{V} \setminus \{v_i\}$.
    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 3 *TGS* algorithm (Graphical flowchart in Figure 2)

procedure TGS($\mathcal{D}$)
  Initialize $\mathcal{G} \leftarrow$ a null graph over $(V \times T)$ nodes.
  $\mathcal{G}_{\text{CLR}} \leftarrow \text{CLR}(\mathcal{D})$. $\triangleright\ \mathcal{O}\left(V^2\right)$
  for each time interval $\left(t_p, t_{(p+1)}\right)$ where $1 \leq p \leq (T-1)$ **do** $\triangleright\ (T-1)$ iterations
    for each gene $v_j \in \mathcal{V}$ **do** $\triangleright\ (V)$ 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 $v_j_t_{(p+1)} \leftarrow powerset\left(\{v_i_t_p : (v_i, v_j) \in \text{Edgeset}(\mathcal{G}_{\text{CLR}})\}\right)$.
      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 *Bene* algorithm. Once the regulator set is finalized, for each node in it, add an edge in $\mathcal{G}$ (Figure 1) from that node to $v_j_t_{(p+1)}$. $\triangleright\ o\left(M^2 2^{(M-2)}\right)$ assuming that M = maximum number of neighbours any gene has in $\mathcal{G}_{\text{CLR}}$ [4, Section 5].
    end for
  end for
  return $\mathcal{G}$.
end procedure

TGS (Algorithm 3) has the time complexity $T_{\text{TGS}}(V) = \left(\mathcal{O}\left(V^2\right) + (T-1) \times V \times o\left(M^2 2^{(M-2)}\right)\right) = \left(\mathcal{O}\left(V^2\right) + o\left((T-1)\, V M^2 2^{(M-2)}\right)\right)$. Here, M is the maximum number of neighbours a gene has in the *CLR* network. Since, in theory, $M \leq V$, time complexity of *TGS* is upper bounded by that of *TBN*. But empirically, it is found [16, 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 [17]) have variants that allow the user to specify the maximum number of regulators a gene can have in a given dataset, known as the max fan-in value (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 max fan-in restriction (Algorithm 4). 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.

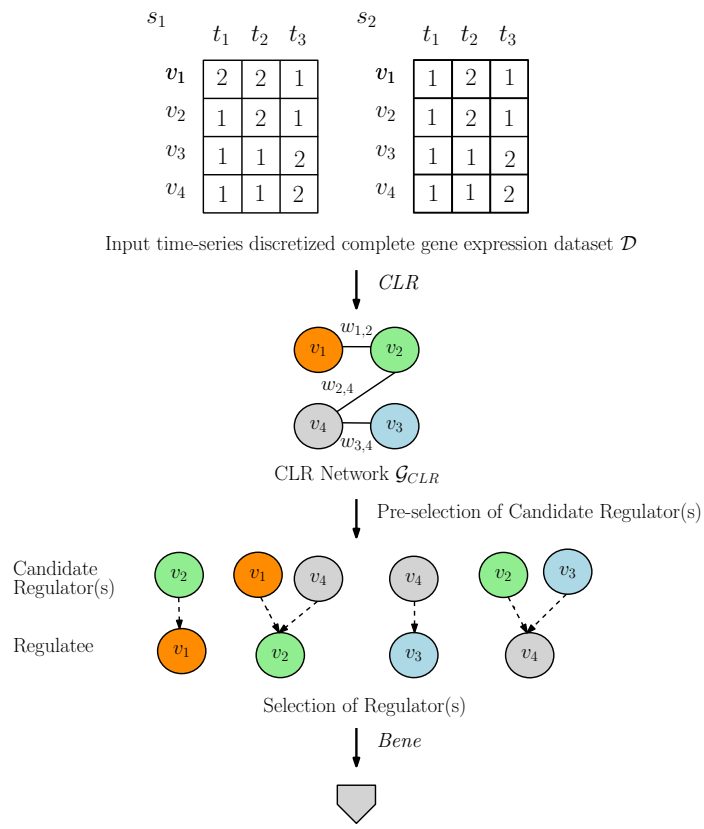


Fig. 2. Graphical Flowchart (Part 1) of the *TGS* Algorithm (Algorithm 3). The flowchart is continued in Figure 3. For illustration, a dataset $\mathcal{D}$ is considered with four genes $\{v_1, v_2, v_3, v_4\} = \mathcal{V}$ and two time series $\{S_1, S_2\} = \mathcal{S}$. Each time series has three time points $\{t_1, t_2, t_3\} = \mathcal{T}$. $\mathcal{D}$ is discretized into two discrete levels, represented by $\{1, 2\}$.

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

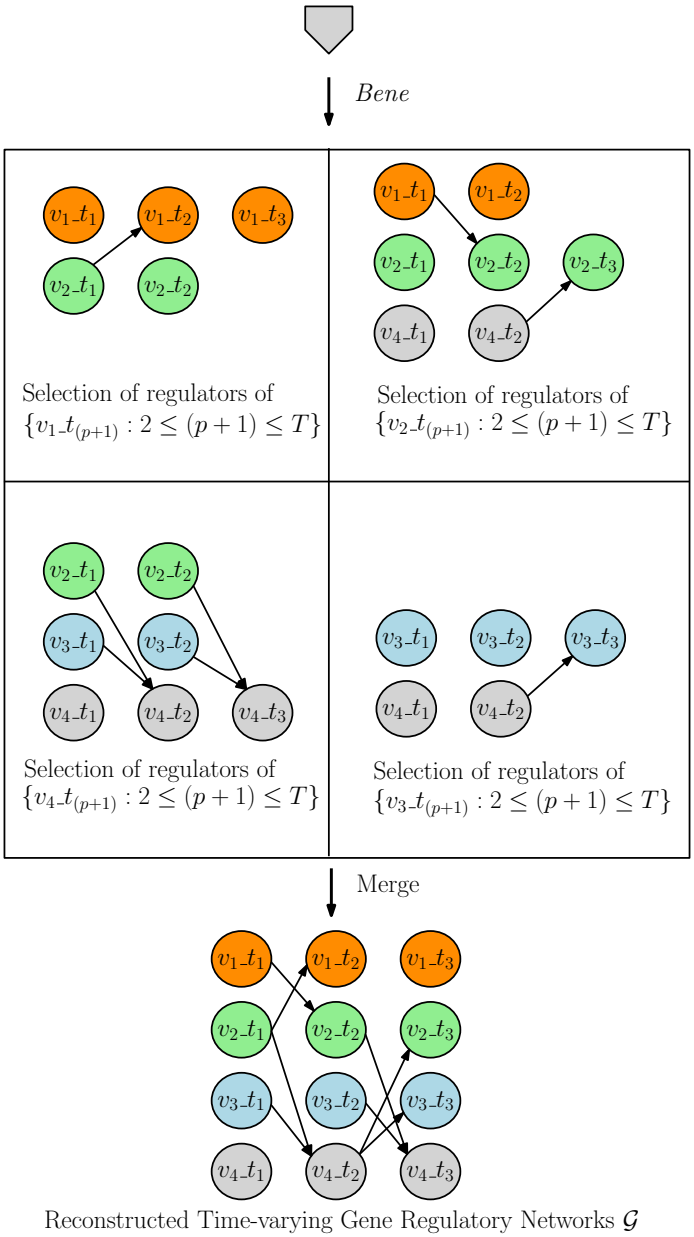


Fig. 3. Graphical Flowchart (Part 2) of the *TGS* Algorithm (Algorithm 3). The flowchart is continued from Figure 2. For discussion of the *Bene* step, let us consider the 'Selection of regulators of $\{v_1_t_{(p+1)} : 2 \leq (p+1) \leq T\}$'. Since, v_2 is the sole neighbour of v_1 in G_{CLR}, v_2 is the only candidate regulator of v_1 (Figure 2). 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 $\mathcal{D}_{\mathcal{V};\{t_2,t_3\};\mathcal{S}}$.

$$
\begin{aligned}
&= \left(\mathcal{O}\left(V^2\right) + o\left(o\left(V^2\left(\lg V\right)^2\right)\left(o\left(V\right)\right)\right)\right) \\
&= \left(\mathcal{O}\left(V^2\right) + o\left(V^3\left(\lg V\right)^2\right)\right) \\
&= o\left(V^3\left(\lg V\right)^2\right) \qquad (2)
\end{aligned}
$$

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

Algorithm 4 *TGS* algorithm with Max fan-in restriction

procedure TGS($\mathcal{D}, \mathcal{M}_{\{}$)
  Initialize $\mathcal{G} \leftarrow$ a null graph over $(V \times T)$ nodes.
  $\mathcal{G}_{\text{CLR}} \leftarrow \text{CLR}\left(\mathcal{D}\right)$. ▷ $\mathcal{O}\left(V^2\right)$
  for each time interval $\left(t_p, t_{(p+1)}\right)$ where $1 \leq p \leq (T-1)$ **do** ▷ $(T-1)$ iterations
    for each gene $v_j \in \mathcal{V}$ **do** ▷ (V) iterations
      if $\left|\{v_i_t_p : (v_i, v_j) \in \text{Edgeset}\left(\mathcal{G}_{\text{CLR}}\right)\}\right| > M_f$ **then**
        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 $v_j_t_{(p+1)} \leftarrow \{v_i_t_p : ((v_i, v_j) \in \text{Edgeset}\left(\mathcal{G}_{\text{CLR}}\right)) \wedge (v_i \in L_j)\}$.
      else
        Candidate regulators of $v_j_t_{(p+1)} \leftarrow \{v_i_t_p : (v_i, v_j) \in \text{Edgeset}\left(\mathcal{G}_{\text{CLR}}\right)\}$.
      end if
      Candidate regulator sets of $v_j_t_{(p+1)} \leftarrow powerset\left(\{v_i_t_p : (v_i, v_j) \in \text{Edgeset}\left(\mathcal{G}_{\text{CLR}}\right)\}\right)$.
      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 *Bene* algorithm. Once the regulator set is finalized, for each node in it, add an edge in $\mathcal{G}$ (Figure 1) 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

4 Results

The results of the *TGS* algorithm on a set of realistically simulated benchmark synthetic datasets and a real dataset are presented in this section. First, *TGS*'s learning power and speed are evaluated with the synthetic datasets, against the baseline algorithm and *ARTIVA*. The baseline algorithm is chosen for the comparative study to analyse the effect of the *CLR* step. Then *TGS* is applied on the real dataset and biological significance of its predictions are evaluated against the existing biological knowledge.

4.1 Datasets

4.1.1 Synthetic DREAM3 In Silico Network Inference Challenge 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. [18] 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 [19, p. 6290, Section 'Simulation of Expression Data.';

TABLE 2
A Summary of the chosen DREAM3 Datasets

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

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 [20], [21]. 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 paper. 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 2 . It can be noted that the true networks are single network GRNs.

4.1.2 Real Drosophila melanogaster (Dm) Life Cycle Dataset (DmLc)

The DmLc dataset is a time series real gene expression dataset of the fruit fly's developmental cycle (from embryonic stage up to the first thirty days of adulthood). This dataset is experimentally produced by Arbeitman et al. [22]. In Song et al. [23, Section 3.2], the dataset is utilized for evaluating performance of the *KELLER* algorithm. Since, the original dataset is real-valued and *KELLER* requires a discretized dataset, the original dataset is discretized using the *2L.Tesla* algorithm (Algorithm 2, Section 4.2, supplementary document), described in Section 4.2). The discretized DmLc dataset is obtained from the *KELLER* website [24]. It contains a single time series ($S = 1$) observations of 4028 ($V = 4028$) genes which represent around one-third of identified Dm genes [22, p. 2271]. There are a total of 66 time points ($T = 66$) in the time series, spread across four life cycle stages; they are the Embryonic (E; time points 1-30), Larval (L; time points 31-40), Pupal (P; time points 41-58) and Adult (A; time points 59-66) stages. The sampling intervals vary between different stages and even within a stage. For example, the sampling intervals are either 0.5 hour or 1 hour in the E stage; whereas, it is in order of days in A stage. This is done because the developmental changes happen at different rates in different stages, like - in the E stage, changes take place faster than those in the A stage. The sampling intervals are described in Arbeitman et al. [22, Fig. 1A]. Following Song et al. [23, p. i132, Section 3.2], a sub-dataset (hereafter, DmLc3) is also generated from the DmLc dataset by selecting the data corresponding to only a subset of 588 genes, known to be involved in the developmental process of Dm according to their gene ontology annotations. Then the DmLc3 dataset is further divided into four mutually exclusive and collectively exhaustive subdatasets with respect to four separate stages. These datasets are appropriately named as DmLc3E, DmLc3L, DmLc3P and DmLcA.

4.2 Discretization of the Datasets

Two different data discretization algorithms are chosen: one (*2L.wt* (Algorithm 1, Section 4.2, Supplementary document)) based on wild type (WT) values of the genes provided through the DREAM challenge in Yeast1 steady state datasets; another (*2L.Tesla* (Algorithm 2, Section 4.2, Supplementary document)) based on a domain-knowledge independent strategy [25, p. 2, Section 'Evolving Gene Networks During Drosophila melanogaster Development. Preprocessing the gene networks.', Supplementary Information].

4.3 Implementation

Both *TBN* and *TGS* are implemented in R programming language [26] version 3.3.2. For *CLR* and *Bene*, their implementations in R packages minet [27] (version: 3.34.0) and bnstruct [28] (version: 1.0.2) are used, respectively. The *ARTIVA* source code is publicly available as an R package with the same name ([29], version: 1.2.3). Experiments are performed on an Intel® computing server. Its configuration is provided in Section 2.2 of the Supplementary document.

4.4 Evaluation Metrics for Comparative Study of the Learning Power

Since the true networks are single network GRNs and the outputs of *TBN* and *TGS* are time-varying GRNs, the output set of networks $\mathcal{G}$ for each algorithm are 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. On the other hand, for Dm datasets, self-loops (if any) are not removed from G as it is not known a prior whether the true network contains any self-loop or not. The metrics used to evaluate correctness of each predicted (rolled) network w.r.t. the corresponding true network are described in Section 4.3 of the supplementary document.

4.5 Learning From Dataset Ds10n

Dataset Ds10n is chosen over Ds10 for comparing *ARTIVA*, *TBN* and *TGS* 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* (Table 3) in terms of learning power (except in TN, FP, FPR and ACC) and speed. Amongst *TBN* and *TGS*, it is found that *TGS* has faster learning speed, which is expected, since the regulator search space for each gene, in case of *TGS*, is monotonically smaller than that of *TBN*.

TABLE 3
Comparative Performance of *ARTIVA*, *TBN* and *TGS* for Learning from Dataset Ds10n. Two ordered values in each cell, under the columns 'TBN' and 'TGS', represent application of two different data discretization algorithms *2L.wt* and *2L.Tesla*, respectively. On the other hand, the 'ARTIVA' column has a single value in each cell, since *ARTIVA* does not require the dataset to be discretized. In *TGS*, the *CLR* step takes 0.003 seconds for each of *2L.wt* and *2L.Tesla*. The numerical values are rounded off to three decimal places.

Evaluation Metric	*ARTIVA*	*TBN*	*TGS*
TP	0	$(3, 1)$	$(3, 2)$
TN	81	$(73, 65)$	$(80, 78)$
FP	9	$(17, 25)$	$(10, 12)$
FN	10	$(7, 9)$	$(7, 8)$
TPR	0	$(0.300, 0.100)$	$(0.300, 0.200)$
FPR	0.1	$(0.189, 0.278)$	$(0.111, 0.133)$
FDR	1	$(0.850, 0.962)$	$(0.769, 0.857)$
PPV	0	$(0.150, 0.038)$	$(0.231, 0.143)$
ACC	0.81	$(0.760, 0.660)$	$(0.830, 0.800)$
MCC	-0.105	$(0.083, -0.122)$	$(0.168, 0.058)$
F1	0	$(0.200, 0.056)$	$(0.261, 0.167)$
Runtime (sec)	619.857	$(7.119, 6.867)$	$(5.789, 5.760)$

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 TPR of *CLR* step is utilized 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 discretization of input gene expression data based on domain-specific knowledge (as in wild type values of genes) improves learning compared to domain-independent alternatives.

4.6 Learning From Datasets Ds50n and Ds100n

Due to *Bene*'s main memory requirement of $2^{(V+2)}$ Bytes [4, 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 fault. 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). But *TBN* does not have any such provision and hence can not be applied on these datasets. Therefore, only *ARTIVA* and *TGS* can be compared for the current study.

In this study, *ARTIVA* consistently outperforms *TGS* in TN and FP, with considerable margins (Table 4). Since the true GRNs are believed to be sparse in nature, it is expected

TABLE 4
Comparative Performance of *ARTIVA* and *TGS* for Learning from Datasets Ds50n and Ds100n. In *TGS*, the *CLR* step takes 0.005 and 0.013 seconds for Ds50n and Ds100n, respectively. Algorithm *2L.wt* is used for data discretization. The numerical values are rounded off to three decimal places.

Evaluation Metric	*Ds50n*		*Ds100n*	
	ARTIVA	*TGS*	*ARTIVA*	*TGS*
TP	6	15	14	28
TN	2359	2081	9676	9044
FP	64	342	158	790
FN	71	62	152	138
TPR	0.078	0.195	0.084	0.169
FPR	0.026	0.141	0.016	0.080
FDR	0.914	0.958	0.919	0.966
PPV	0.086	0.042	0.081	0.034
ACC	0.946	0.838	0.969	0.907
MCC	0.054	0.026	0.067	0.041
F1	0.082	0.069	0.083	0.057
Runtime (sec)	16214.996	455.764	114774.090	1069.100

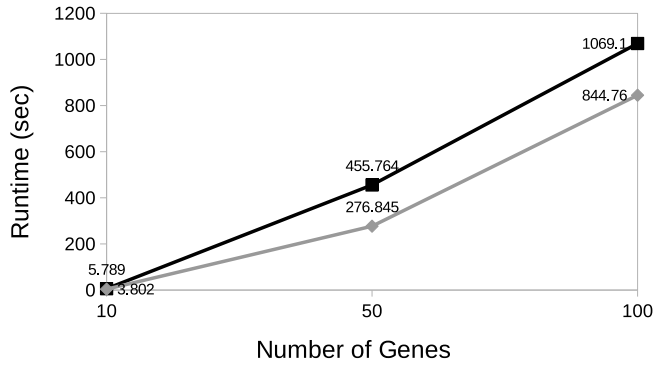


Fig. 4. Runtime of the *TGS* Algorithm w.r.t. the Number of Genes in the Input Synthetic Datasets (Table 2). The black and grey lines represent noisy and noiseless versions of the datasets, respectively.

that, among the 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 higher FN and considerably lower TP, compared to *TGS*. Another major concern with *ARTIVA* is the runtime. It takes 114774.090 seconds ($\simeq$ 32 hours) to reconstruct 100-gene GRNs, which is certainly a bottleneck for its application in reconstructing human genome-scale GRNs. In comparison, *TGS* consumes only 1069.100 seconds ($\simeq$ 18 minutes). Moreover, *TGS*'s runtime grows almost linearly as the number of genes grow (Figure 4). These observations indicate that *TGS* is substantially more suitable for reconstructing large-scale GRNs than *ARTIVA*.

4.7 Effect of Noise on Learning Power and Speed

TGS is evaluated on all noisy and noiseless datasets with different number of genes. From Figures 4 and 5, it can

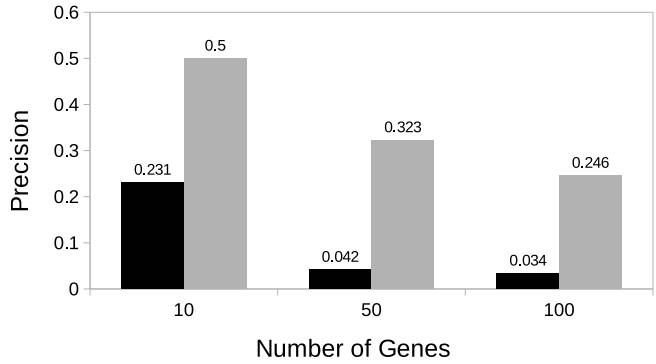


Fig. 5. Precision of the *TGS* Algorithm w.r.t. the Number of Genes in the Datasets. The black and grey bars represent noisy and noiseless versions of the datasets, respectively. The *2L.wt* algorithm is used for data discretization.

TABLE 5
Maximum Number of Neighbours a Gene has in the *CLR* Network. Algorithm *2L.wt* is used for data discretization.

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

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). In the absence of noise, the *CLR* step can eliminate more number 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.

4.8 Learning from the DmLc3 Datasets

For each sub-dataset of the DmLc3 dataset, all time points belong to a single time series. However, *TGS* requires multiple time series as input. Hence, the time points are divided into multiple groups and each group is considered as a distinct time series. For example, DmLcE is originally comprised of 30 time points belonging to a single time series. These time points are used to generate 5 time series having 6 time points each, with the following strategy: time points $1-5$ are assigned to time series $1-5$, respectively; similarly, time points $6-10$ are assigned to time series $1-5$, respectively; and so on. In short, the ith time point is assigned to the $(i \mod S)^{th}$ time series, where i varies from 1 to the total number of time points in the original time series and S equals the number of time series to be generated. This strategy ensures that the replicates at each newly created time point are consecutive time points in the original time series. For example, there are 5 replicates at the newly created first time point. It comprises of the first time points of every newly created time series. Therefore, these replicates are time points $1-5$ in the original time series. The same single to multiple time series conversion strategy is followed for datasets DmLcL, DmLcP and DmLcA (Table 6).

TABLE 6
A Summary of the DmLc3 Datasets before and after the Conversion from Single to Multiple Time Series. The datasets, after conversion, are used for experimentation. Here, V = number of genes, T = number of time points and S = number of time series in a given dataset.

Dataset	Before			After		
	V	T	S	V	T	S
DmLc3E	588	30	1	588	6	5
DmLc3L	588	10	1	588	2	5
DmLc3P	588	18	1	588	3	6
DmLc3A	588	8	1	588	2	4

The *TGS* algorithm is applied separately on each converted dataset. It results in reconstruction of $(T-1)$ time-varying GRN(s) (one GRN for each time interval) for each dataset, where T = number of time points in that dataset. Therefore, five time-varying GRNs $\{G_E^{(1)}, G_E^{(2)}, \cdots, G_E^{(5)}\}$ are reconstructed from DmLc3E; also a single time-invariant GRN G_E is generated by rolling up the time-varying GRNs. For DmLc3L, one time-varying GRN G_L is reconstructed; since there is only one time-varying GRN, there is no need to roll it up. For DmLc3P, two time-varying GRNs $\{G_P^{(1)}, G_P^{(2)}\}$ are reconstructed; also a single time-invariant GRN G_P is generated by rolling up the time-varying GRNs. For DmLc3A, one time-varying GRN G_A is reconstructed; again since there is only one time-varying GRN, there is no need to roll it up. The accuracy of all the reconstructed time-varying GRNs and the time taken to reconstruct them are studied in the subsequent sections.

4.8.1 Study of Learning Power with the DmLc3 Datasets

The study is conducted in two steps. In the first step, a coarse-grained analysis is performed at the network level. It analyses structural properties of the predicted GRNs. In the second step, a fine-grained analysis is performed at the edge level. It attempts to evaluate the biological relevance of the predicted edge relationships. Since, the true GRNs are not known, the evaluation is performed against the existing biological knowledge. The outcomes of these two steps are described below.

Coarse-grained Analysis: For the coarse-grained analysis, Cytoscape [30], [31], [32] is used to compute network statistics of the predicted networks (Table 7). The following observations can be made from the network statistics:

(A) The network parameters have higher variability among the stage-specific GRNs (Table 7a) than amongst the intra-stage GRNs (Table 7b). For example, in case of stage-specific GRNs, the bias-corrected standard deviation [33] of the 'Number of edges' row is 566.4495. It is much higher than that (115.1173) of intra-stage GRNs.

(B) From Table 7a, it can be observed that G_E and G_P are topologically more similar to each other, while G_L and G_A are topologically more similar to each other. In Arbeitman et al. [22], the same similarities are observed at the gene expression levels; the hierarchical clustering of the DmLc dataset's time points groups L and A stage time points

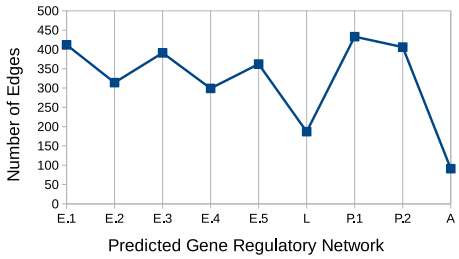
TABLE 7
Statistics of the Predicted Gene Regulatory Networks

Parameter	$\mathbf{G_E}$	$\mathbf{G_L}$	$\mathbf{G_P}$	$\mathbf{G_A}$
Number of edges	1309	187	775	91
Number of self-loops	8	17	2	0
Clustering coefficient	0.077	0	0.017	0
Number of connected components	11	418	39	497
Number of isolated nodes	10	408	35	492
Network diameter	29	8	22	2
Characteristic path length	10.413	2.626	8.432	1.248
Average number of neighbours	4.184	0.578	2.565	0.31
Number of length 2 cycles	71	0	19	0
Maximum in-degree (Number of such nodes)	5 (2)	1 (187)	2 (235)	1 (91)
Minimum in-degree (Number of such nodes)	0 (18)	0 (401)	0 (48)	0 (497)
Mode of in-degree (Number of such nodes)	2 (241)	0 (401)	1 (305)	0 (497)
Maximum out-degree (Number of such nodes)	21 (1)	14 (1)	24 (2)	50 (1)
Minimum out-degree (Number of such nodes)	0 (259)	0 (538)	0 (380)	0 (580)
Mode of out-degree (Number of such nodes)	0 (259)	0 (538)	0 (380)	0 (580)
Appearance of out-degree distribution	Power law	Power law	Power law	Dis-continuous

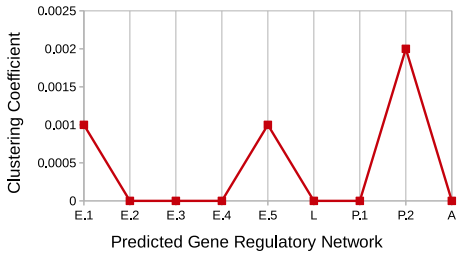
(a) Statistics of the Stage-specific Gene Regulatory Networks

Parameter	$\mathbf{G_E^{(1)}}$	$\mathbf{G_E^{(2)}}$	$\mathbf{G_E^{(3)}}$	$\mathbf{G_E^{(4)}}$	$\mathbf{G_E^{(5)}}$	$\mathbf{G_L}$	$\mathbf{G_P^{(1)}}$	$\mathbf{G_P^{(2)}}$	$\mathbf{G_A}$
Number of edges	412	314	391	299	362	187	433	406	91
Number of self-loops	0	3	4	0	1	17	1	1	0
Clustering coefficient	0.001	0	0	0	0.001	0	0	0.002	0
Number of connected components	186	284	210	293	237	418	160	191	497
Number of isolated nodes	152	243	175	260	208	408	137	163	492
Network diameter	8	8	8	8	8	8	14	9	2
Characteristic path length	2.451	2.323	2.135	1.974	2.410	2.626	3.389	2.457	1.248
Average number of neighbours	1.378	1.034	1.289	1.003	1.197	0.578	1.456	1.354	0.310
Number of length 2 cycles	7	7	8	4	9	0	4	7	0
Maximum in-degree (Number of such nodes)	1 (412)	1 (314)	1 (391)	1 (299)	1 (362)	1 (187)	1 (433)	1 (406)	1 (91)
Minimum in-degree (Number of such nodes)	0 (176)	0 (274)	0 (197)	0 (289)	0 (226)	0 (401)	0 (155)	0 (182)	0 (497)
Mode of in-degree (Number of such nodes)	1 (412)	1 (314)	1 (391)	1 (299)	1 (362)	0 (401)	1 (433)	1 (406)	0 (497)
Maximum out-degree (Number of such nodes)	18 (1)	14 (1)	16 (1)	14 (1)	20 (1)	14 (1)	23 (1)	24 (1)	50 (1)
Minimum out-degree (Number of such nodes)	0 (444)	0 (456)	0 (452)	0 (484)	0 (462)	0 (538)	0 (451)	0 (469)	0 (580)
Mode of out-degree (Number of such nodes)	0 (444)	0 (456)	0 (452)	0 (484)	0 (462)	0 (538)	0 (451)	0 (469)	0 (580)
Appearance of out-degree distribution	Power law	Power law	Power law	Power law	Power law	Power law	Power law	Power law	Dis-continuous

(b) Statistics of the Time Interval Specific (intra-stage) Gene Regulatory Networks



(a) Number of Edges vs Predicted Gene Regulatory Networks



(b) Clustering Coefficients vs Predicted Gene Regulatory Networks

Fig. 6. Number of Edges and Clustering Coefficients of the predicted Time-varying Gene Regulatory Networks (GRNs). $E.1$, $E.2$, $E.3$, $E.4$, $E.5$, L, $P.1$, $P.2$ and A are the shortened forms for the GRNs $G_E^{(1)}$, $G_E^{(2)}$, $G_E^{(3)}$, $G_E^{(4)}$, $G_E^{(5)}$, G_L, $G_P^{(1)}$, $G_P^{(2)}$ and G_A, respectively.

in one cluster, and P and E stage time points in another [22, Fig. 1 (F)]. Moreover, the genes that are expressed in the early E stage, are often re-expressed in the early P stage and the genes that expressed in the late E stage, are often re-expressed in the late P stage [22, Fig. 1 (E)]. Such commonality in the expression levels may have arisen from the presence of common regulatory events during the E and P stages, as it is found that G_E and G_P share a total of 98 common edges.

(C) G_E and G_P have more number of edges than those of G_L and G_A (Table 7a). More specifically, number of edges peaks once at the beginning of the E stage and then again at the beginning of the P stage (Figure 6a). This prediction matches with the *KELLER*'s prediction [23, Network Size (NS) curve in Fig. 3(a)]. Additionally, it is observed that $G_E^{(1)}$ and $G_P^{(1)}$ have the least number of isolated nodes, compared to other intra-stage GRNs (Table 7b). Therefore, one plausible explanation can be hypothesized that higher numbers of genes are involved in the developmental processes during the beginnings of the E and the P stages, compared to other time intervals.

(D) The clustering coefficient drops sharply as the E stage progresses (Figure 6b), which is similar to the prediction made by *KELLER* [23, Clustering Coefficient (CC) curve in Fig. 3(a)]. But then the coefficient rises at the end of the E stage before descending again at the L stage, whereas in *KELLER*'s prediction, it remains plateaued till the beginning of the A stage. Finally, the clustering coefficient makes another ascent just before the A stage begins, while in case of *KELLER*, it does so after the A stage begins.

Two predictions that *TGS* and *KELLER* both agree on are: **(i)** the number of edges is the highest and the clustering coefficient is the lowest in beginning of the P stage i.e. the start of the metamorphosis, and **(ii)** although, the number of edges in beginning of the E stage is as high as that in P stage's beginning, the clustering coefficient of the former is much higher than that of the latter. The reason behind this observation is that $G_E^{(1)}$ possesses higher number of feedback loops of length three than $G_P^{(1)}$.

(E) G_E and G_P seem to have small numbers of large connected components, whereas G_L and G_A appear to have large numbers of small connected components. It is reflected by the rows 'Number of connected components', 'Network diameter' and 'Characteristic path length' in Table 7a.

(F) In G_E, most of the genes have multiple regulators (in-degree > 1), whereas in other stage-specific networks, most of them do not (Row 'Mode of in-degree', Table 7a). However, this variability disappears in the intra-stage statistics, where no gene has more than one regulator at any time interval (Row 'Mode of in-degree', Table 7b). One argument can be put forward that a gene is not necessarily regulated by all its regulators at the same time interval; it may be regulated by different regulators at different time intervals.

(G) The out-degree distributions of all the predicted intra-stage and stage-specific GRNs, except G_A, follow the power law (Rows 'Appearance of out-degree distribution', Table 7), which is a common property in many large real networks [34]. There are a handful of genes that regulates most of the genes. It is a common phenomenon in the biological networks where such genes are loosely termed as the master regulators [35], [36]. Due to such a phenomenon, true GRNs tend to be sparse [11, Section 'Problems with GGMs']. In agreement, all the predicted stage-specific GRNs are found to be sparse i.e. their network densities [2] vary from 0.0003 (for G_A) to 0.0038 (for G_E).

(H) The predicted stage-specific and intra-stage GRNs in the E and P stages have positive numbers of length 2 cycles (Rows 'Number of length 2 cycles', Table 7). A length 2 cycle indicates that two genes are cyclically regulating each other. Cyclic regulatory relationship is a common characteristic of the biological networks [11, Section 'Acyclicity in a cyclic world'].

For the complete set of findings of the coarse-grained analysis, please see Table 3.1 of the supplementary document.

Fine-grained Analysis: For the fine-grained analysis, a subset of 25 genes are chosen, which are known to produce TFs in Dm. This subset is generated by intersecting the set of genes in the DmLc3 dataset with the set of known TF-coding Dm genes used in Marbach et al. [37]. Then for each gene in the subset, two questions are posed:

Q1. Whether the given gene is predicted to play any regulatory role in the development stage(s) where it is known to do so? This question is answered by checking whether the gene has at least one regulatee in the predicted GRN(s) specific to that stage(s).

Q2. If answer to Q1 is yes, then does the given gene regulate any of its known regulatees (if any) in the predicted GRN(s)? This question is inapplicable when answer to Q1 is no.

For this analysis, known regulatory stages and known regulatees, if any, of each concerned gene are retrieved from TRANSFAC Public Database version 7.0 [38], which is claimed to be the gold standard in the area of transcriptional regulation [39]. This analysis finds a number of biological supports for the predicted GRNs. Some of the findings are discussed below.

Gene 'prd' is known to have a positive cell specificity in the E stage. It is also known that 'prd' participates in the regulation of anterior-posterior segmentation of the embryo. In agreement, 'prd' is predicted to have the maximum number (8) of regulatees in G_E, whereas it has 3 regulatees in G_P, and does not have any regulatee in G_L and G_A. Additionally, in G_E, 'prd' regulates 'eve', which is a known regulatee of 'prd'.

Similar to 'prd', 'bcd' is known to be a major regulator in the anterior-posterior axis formation of the embryo. The prediction is consistent with the knowledge as 'bcd' has 9 regulatees in G_E and does not possess any regulatee in either of $\{G_L, G_P, G_A\}$. 'bcd' is also known to be a concentration-dependent regulator of 'eve'. But there is no directed edge from 'bcd' to 'eve' in G_E. It might be a true negative prediction if the regulation did not happen during the data collection period owing to the absence of the required concentration level.

2. Network density is computed as the ratio of the number of edges to the maximum possible number of edges. Here, the maximum possible number of edges in a directed network is calculated as the square of the number of nodes (genes) in the network.

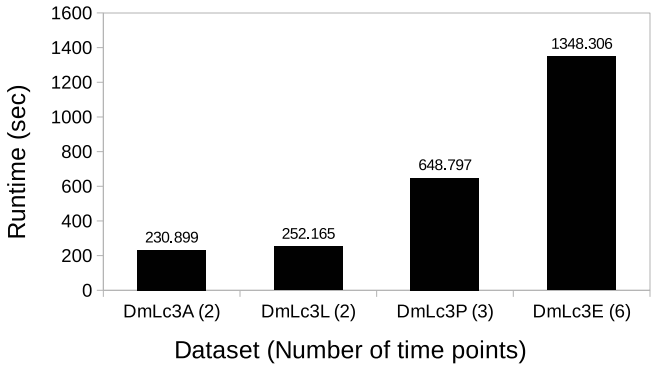


Fig. 7. Runtime of the *TGS* algorithm w.r.t. DmLc3 datasets. These datasets (Table 6) have the same number of genes ($V = 588$). But they vary in the number of time points (T). It is evident that the runtime strictly increases with T when V is fixed, which is consistent with the time complexity expression in Equation 1.

Some more genes, like - 'tll', 'dl', 'ftz.f1' and 'Trl' are reported in literature to play regulatory roles in the E stage. In accordance with the literature, they are found to have no regulatee in any predicted networks except in G_E. Moreover, 'Trl' produces a very abundant nuclear protein, known as the GAGA protein, that has implications in the transcriptions of numerous Dm genes by either directly binding to the regulatee gene's TF binding site or by allowing the regulatee gene to open up for transcription via modification of the chromatin configuration around it [40]. This implication is also found in G_E, where 'Trl' is predicted to have directed paths to a total of 538 genes (downstream regulatees) in spite of having directed edges to only 2 genes (direct regulatees).

Another interesting prediction is found for gene 'Antp'. Appel et al. [41] propose that, in some type of neuronal cells, the regulator proteins of 'Antp' compete to regulate it with the protein encoded by 'Antp' itself; in other words, 'Antp' appears to auto-regulate itself. But, in none of the predicted GRNs, 'Antp' has a self-loop. However in G_E, it does have two feedback loops (directed paths to itself), each of length 2, through genes 'odd' and 'CG12896', respectively. Whether the auto-regulation of 'Antp' happens through a self-loop or multi-hop feedback loops opens an intriguing question to answer in experimental biology.

For the complete set of findings of the fine-grained analysis, please see Table 3.1 of the supplementary document.

4.8.2 Study of Learning Speed with the DmLc3 Datasets

The comparative study of DmLc3 datasets reveals that the runtime of the *TGS* algorithm increases with the value of T, when V is fixed (Figure 7). This finding is consistent with the time complexity expression in Equation 1. The contribution of the *CLR* step in the runtime remains negligible, 0.830 seconds being the maximum for the DmLc3E dataset.

An additional experiment is performed to examine whether *TGS* can scale up to larger datasets than the DmLc3 datasets in a reasonable time frame. For this experiment, the expression levels of all 4028 genes of the DmLc dataset during the E stage are used. The E stage consists of 30 time points belonging to a single time series. They are converted to 15 time series, each consisting of 2 time points using the strategy discussed in Section 4.8. The resultant dataset is named DmLcE. *TGS* is able to scale up to DmLcE, reconstructing a GRN with 3318 directed edges, in 1712.067 seconds ($\simeq$ 29 minutes). During that time, the *CLR* step takes up only 8.038 seconds.

Detailed guidelines of how to reproduce the results, presented in this section, are provided in the supplementary document.

5 Summary and Future Work

In this paper, a novel algorithm, namely *TGS*, is proposed to reconstruct time-varying GRNs from a time series gene expression dataset. *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. In the first step, for each target gene, a shortlist of its potential regulators is inferred. In the final step, these shortlisted candidates are thoroughly evaluated to identify the true regulators among them. 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 any structural constraint. If there exists an association between two GRN structures in the real system and the data reflects the association, then it is also captured in the reconstructed GRN structures.

However, an existing algorithm, namely *ARTIVA*, is already able to provide a similarly flexible framework [1, Section 'Conclusions']. The only challenge with *ARTIVA* is its substantial runtime. That makes *ARTIVA*'s application prohibitive with high-throughput 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.

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). Nevertheless, *TV-DBN* accomplishes such scalability by imposing an explicit structural constraint, called the smoothly time-varying assumption. It assumes 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* provides the flexibility and time-efficiency without losing its accuracy. It consistently outperforms *ARTIVA* in true positive detection, given three benchmark realistically simulated datasets.

Nevertheless, there are scopes for improvement. One limitation of *TGS* is the need to discretize the gene expression data, which is inherently continuous. The reason behind that is *TGS* uses BIC score to determine the best GRN structure and BIC scoring function requires the data to be discretized. 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 discretized, like in Grzegorczyk et al. [10], and (b) by developing a regression based structure learning strategy, e.g., Lèbre et al. [1], because regression problems are inherently compatible with continuous data.

However, the experiments with the large datasets help the authors 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 the datasets. In the current implementation of *TGS*, maximum number of candidate regulators is restricted to 14 for each gene, to avoid this issue. But relaxing this restriction is an important challenge since the true number of regulators for a gene is not known a prior.

The reason behind such astronomical memory requirement is that *Bene* and related Bayesian Network structure learning algorithms need to compute and store the global conditional probability table [4, 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. [42]. Mending this gap can be considered a worthwhile challenge. 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.

Acknowledgements

S. Pyne is supported by the Ministry of Human Resource Development, Government of India and Indian Institute of Technology Guwahati through the Institute Assistantship programme. A. R. Kumar has received the same support during July, 2015 - July, 2017.

References

[1] S. Lèbre, J. Becq, F. Devaux, M. P. Stumpf, and G. Lelandais, "Statistical inference of the time-varying structure of gene-regulation networks," *BMC Systems Biology*, vol. 4, no. 1, p. 130, Sep 2010. [Online]. Available: https://doi.org/10.1186/1752-0509-4-130

[2] L. Song, M. Kolar, and E. P. Xing, "Time-varying dynamic bayesian networks," in *Advances in Neural Information Processing Systems*, 2009, pp. 1732–1740.

[3] N. Friedman, K. Murphy, and S. Russell, "Learning the structure of dynamic probabilistic networks," in *Proceedings of the Fourteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*. San Francisco, CA: Morgan Kaufmann, 1998, pp. 139–147.

[4] T. Silander and P. Myllymäki, "A simple approach for finding the globally optimal bayesian network structure," in *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, ser. UAI'06. Arlington, Virginia, United States: AUAI Press, 2006, pp. 445–452. [Online]. Available: http://dl.acm.org/citation.cfm?id=3020419.3020473

[5] V. A. Huynh-Thu, A. Irrthum, L. Wehenkel, and P. Geurts, "Inferring regulatory networks from expression data using tree-based methods," *PLOS ONE*, vol. 5, no. 9, pp. 1–10, 09 2010. [Online]. Available: https://doi.org/10.1371/journal.pone.0012776

[6] X. Zhang, K. Liu, Z.-P. Liu, B. Duval, J.-M. Richer, X.-M. Zhao, J.-K. Hao, and L. Chen, "Narromi: a noise and redundancy reduction technique improves accuracy of gene regulatory network inference," *Bioinformatics*, vol. 29, no. 1, pp. 106–113, 2013. [Online]. Available: +http://dx.doi.org/10.1093/bioinformatics/bts619

[7] F. Liu, S.-W. Zhang, W.-F. Guo, Z.-G. Wei, and L. Chen, "Inference of gene regulatory network based on local bayesian networks," *PLOS Computational Biology*, vol. 12, no. 8, pp. 1–17, 08 2016. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1005024

[8] M. Grzegorczyk and D. Husmeier, "Improvements in the reconstruction of time-varying gene regulatory networks: dynamic programming and regularization by information sharing among genes," *Bioinformatics*, vol. 27, no. 5, pp. 693–699, 2011. [Online]. Available: +http://dx.doi.org/10.1093/bioinformatics/btq711

[9] J. W. Robinson and A. J. Hartemink, "Non-stationary dynamic bayesian networks," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2008, pp. 1369–1376. [Online]. Available: http://papers.nips.cc/paper/3571-non-stationary-dynamic-bayesian-networks.pdf

[10] M. Grzegorczyk and D. Husmeier, "Non-stationary continuous dynamic bayesian networks," in *Advances in Neural Information Processing Systems 22*, Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, Eds. Curran Associates, Inc., 2009, pp. 682–690. [Online]. Available: http://papers.nips.cc/paper/3687-non-stationary-continuous-dynamic-bayesian-networks.pdf

[11] F. Markowetz and R. Spang, "Inferring cellular networks – a review," *BMC Bioinformatics*, vol. 8, no. 6, p. S5, Sep 2007. [Online]. Available: https://doi.org/10.1186/1471-2105-8-S6-S5

[12] K. P. Murphy, "How to use the bayes net toolbox," last accessed on Aug 15, 2017. The documentation was last updated on Oct 29, 2007. [Online]. Available: http://bayesnet.github.io/bnt/docs/usage.html

[13] T. Silander, "Hyperparameter sensitivity revisited," in *Advanced Methodologies for Bayesian Networks*, 2017, pp. 7–7.

[14] Z.-P. Liu, "Quantifying gene regulatory relationships with association measures: A comparative study," *Frontiers in Genetics*, vol. 8, p. 96, 2017. [Online]. Available: http://journal.frontiersin.org/article/10.3389/fgene.2017.00096

[15] J. J. Faith, B. Hayete, J. T. Thaden, I. Mogno, J. Wierzbowski, G. Cottarel, S. Kasif, J. J. Collins, and T. S. Gardner, "Large-scale mapping and validation of escherichia coli transcriptional regulation from a compendium of expression profiles," *PLOS Biology*, vol. 5, no. 1, pp. 1–13, 01 2007. [Online]. Available: https://doi.org/10.1371/journal.pbio.0050008

[16] N. Bhardwaj, M. B. Carson, A. Abyzov, K.-K. Yan, H. Lu, and M. B. Gerstein, "Analysis of combinatorial regulation: Scaling of partnerships between regulators with the number of governed targets," *PLOS Computational Biology*, vol. 6, no. 5, pp. 1–9, 05 2010. [Online]. Available: https://doi.org/10.1371/journal.pcbi.1000755

[17] K. P. Murphy, "The bayes net toolbox for matlab," *Computing Science and Statistics*, vol. 33, p. 2001, 2001.

[18] D. Marbach, T. Schaffter, C. Mattiussi, and D. Floreano, "Generating Realistic In Silico Gene Networks for Performance Assessment of Reverse Engineering Methods," *Journal of Computational Biology*, vol. 16, no. 2, pp. 229–239, 2009, wingX.

[19] D. Marbach, R. J. Prill, T. Schaffter, C. Mattiussi, D. Floreano, and G. Stolovitzky, "Revealing strengths and weaknesses of methods for gene network inference," *PNAS*, vol. 107, no. 14, pp. 6286–6291, 2010, wingX.

[20] DREAM3, "Dream3 in silico network challenge," last accessed: May 15, 2017. [Online]. Available: \url{https://www.synapse.org/#!Synapse:syn2853594/wiki/71567}

[21] R. J. Prill, D. Marbach, J. Saez-Rodriguez, P. K. Sorger, L. G. Alexopoulos, X. Xue, N. D. Clarke, G. Altan-Bonnet, and G. Stolovitzky, "Towards a rigorous assessment of systems biology models: The dream3 challenges," *PLOS ONE*, vol. 5, no. 2, pp. 1–18, 02 2010. [Online]. Available: https://doi.org/10.1371/journal.pone.0009202

[22] M. N. Arbeitman, E. E. M. Furlong, F. Imam, E. Johnson, B. H. Null, B. S. Baker, M. A. Krasnow, M. P. Scott, R. W. Davis, and K. P. White, "Gene expression during the life cycle of drosophila melanogaster," *Science*, vol. 297, no. 5590, pp. 2270–2275, 2002. [Online]. Available: http://science.sciencemag.org/content/297/5590/2270

[23] L. Song, M. Kolar, and E. P. Xing, "Keller: estimating time-varying interactions between genes," *Bioinformatics*, vol. 25, no. 12, pp. i128–i136, 2009. [Online]. Available: +http://dx.doi.org/10.1093/bioinformatics/btp192

[24] DmLc, "Drosophila melanogaster life cycle dataset," last accessed:

Oct 7, 2017. [Online]. Available: http://www.sailing.cs.cmu.edu/main/keller/data.zip

[25] A. Ahmed and E. P. Xing, "Recovering time-varying networks of dependencies in social and biological studies," *Proceedings of the National Academy of Sciences*, vol. 106, no. 29, pp. 11 878–11 883, 2009.

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

[27] P. E. Meyer, F. Lafitte, and G. Bontempi, "minet: Ar/bioconductor package for inferring large transcriptional networks using mutual information," *BMC bioinformatics*, vol. 9, no. 1, p. 461, 2008.

[28] A. Franzin, F. Sambo, and B. Di Camillo, "bnstruct: an r package for bayesian network structure learning in the presence of missing data," *Bioinformatics*, p. btw807, 2016. [Online]. Available: https://cran.r-project.org/package=bnstruct

[29] ARTIVA, "ARTIVA package," Last accessed: Oct 13, 2017. [Online]. Available: https://cran.r-project.org/package=ARTIVA

[30] "Cytoscape website," Last accessed: Oct 10, 2017. [Online]. Available: http://www.cytoscape.org/

[31] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker, "Cytoscape: a software environment for integrated models of biomolecular interaction networks," *Genome research*, vol. 13, no. 11, pp. 2498–2504, 2003.

[32] R. Christmas, I. Avila-Campillo, H. Bolouri, B. Schwikowski, M. Anderson, R. Kelley, N. Landys, C. Workman, T. Ideker, E. Cerami, R. Sheridan, G. D. Bader, and C. Sander, "Cytoscape: A software environment for integrated models of biomolecular interaction networks," 2005, pp. 12–16. [Online]. Available: http://educationbook.aacrjournals.org/cgi/content/full/2005/1/12

[33] "Standard Deviation," Equation 5. Last accessed: Dec 1, 2017. [Online]. Available: http://mathworld.wolfram.com/StandardDeviation.html

[34] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *science*, vol. 286, no. 5439, pp. 509–512, 1999.

[35] H. Tovar, R. García-Herrera, J. Espinal-Enríquez, and E. Hernández-Lemus, "Transcriptional master regulator analysis in breast cancer genetic networks," *Computational biology and chemistry*, vol. 59, pp. 67–77, 2015.

[36] S. S.-K. Chan and M. Kyba, "What is a master regulator?" *Journal of stem cell research & therapy*, vol. 3, 2013.

[37] D. Marbach, S. Roy, F. Ay, P. E. Meyer, R. Candeias, T. Kahveci, C. A. Bristow, and M. Kellis, "Predictive regulatory models in drosophila melanogaster by integrative inference of transcriptional networks," *Genome research*, vol. 22, no. 7, pp. 1334–1349, 2012.

[38] B. GmbH, "Transfac public database version 7.0," the user requires to create a free-of-cost account to access the database. Last accessed: Oct 10, 2017. [Online]. Available: http://gene-regulation.com/cgi-bin/pub/databases/transfac/search.cgi

[39] "Genexplain 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).". [Online]. Available: http://genexplain.com/transfac/

[40] T. O'Brien, R. C. Wilkins, C. Giardina, and J. T. Lis, "Distribution of gaga protein on drosophila genes in vivo." *Genes & development*, vol. 9, no. 9, pp. 1098–1110, 1995.

[41] B. Appel and S. Sakonju, "Cell-type-specific mechanisms of transcriptional repression by the homeotic gene products ubx and abd-a in drosophila embryos." *The EMBO journal*, vol. 12, no. 3, p. 1099, 1993.

[42] N. Jahnsson, B. Malone, and P. Myllymäki, "Duplicate detection for bayesian network structure learning," *New Generation Computing*, vol. 35, no. 1, pp. 47–67, Jan 2017. [Online]. Available: https://doi.org/10.1007/s00354-016-0004-9

Saptarshi Pyne received the BTech degree from the Department of Computer Science and Engineering, Future Institute of Engineering and Management, India, in 2010. He is a PhD student in Dr. Ashish Anand's research group. Saptarshi believes in a future where biomolecular signals are measured in-vivo and analysed in near real time. **LinkedIn:** https://in.linkedin.com/in/saptarshi-pyne-5ba76916

Alok Ranjan Kumar received the BE degree from the Department of Computer Science and Engineering, Birla Institute of Technology, India, in 2012, and the MTech degree from the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati, India, in 2017. He wrote his MTech thesis, titled 'Structure learning of gene regulatory network from large-scale (high-dimensional) time-series gene expression data', under the supervision of Dr. Ashish Anand. Alok is a graduate trainee engineer at Siemens Industry Software (India) Pvt. Ltd. **LinkedIn:** https://www.linkedin.com/in/alok-kumar-4039/.

Ashish Anand received the Masters (Int-MSc, 5 years) degree in Mathematics and Scientific Computing from Indian Institute of Technology Kanpur, India, in 2002, and the PhD degree from Nanyang Technological University, Singapore, on 'Computational Intelligence Methods for Problems in Computational Biology', in 2009. He is an assistant professor at the Department of Computer Science and Engineering, Indian Institute of Technology Guwahati (IITG), India. Prior to joining IITG, he was part of the European Consortium, BaSySBio at the Systems Biology Lab, Institut Pasteur, Paris. His post-doc work was primarily concentrated on regulatory network reconstruction and pathway analysis. His main research area is temporal progression modelling of biological systems. **Website:** http://iitg.ac.in/anand.ashish/