bLARS: An Algorithm to Infer Gene Regulatory Networks

Nitin Singh and Mathukumalli Vidyasagar

Abstract—Inferring gene regulatory networks (GRNs) from high-throughput gene-expression data is an important and challenging problem in systems biology. Several existing algorithms formulate GRN inference as a regression problem. The available regression based algorithms are based on the assumption that all regulatory interactions are linear. However, nonlinear transcription regulation mechanisms are common in biology. In this work, we propose a new regression based method named bLARS that permits a variety of regulatory interactions from a predefined but otherwise arbitrary family of functions. On three DREAM benchmark datasets, namely gene expression data from E. coli, Yeast, and a synthetic data set, bLARS outperforms state-of-the-art algorithms in the terms of the *overall score*. On the individual networks, bLARS offers the best performance among currently available similar algorithms, namely algorithms that do not use perturbation information and are not meta-algorithms. Moreover, the presented approach can also be utilized for general feature selection problems in domains other than biology, provided they are of a similar structure.

Index Terms—Gene regulatory network, reverse engineering algorithms, DREAM challenge, regression

1 Introduction

The complex interaction of genes and their products regulates the behavior of living cells. The knowledge of these regulatory interactions is critical to understand cellular processes and disease mechanisms. Collectively these interactions are very often referred to as Gene Regulatory Networks (GRNs). The advent of high-throughput techniques presents an opportunity for the construction of GRNs in a systematic fashion. As a result, numerous GRN inference algorithms have been invented in the past decade; see Section 1.1. These algorithms can be categorized by the techniques that are employed to formulate and solve the GRN inference problem [1]. Some of these categories and their representative algorithms are: mutual information (CLR [2], ARACNE [3]), regression (TIGRESS [4]), graphical models (Bayesian Networks[5]), correlation (Relevance Network [6]), ensemble methods (GENIE3 [7]) and others (ANOVA η^2 [8]). Among these, the regression class of algorithms assume that the gene expression level of a target gene is a linear combination of the expression levels of its transcription factors. In subsequent steps, a small set of transcription factors is selected, whose linear combination best explains the expression of the target gene, using sparsity inducing methods such as l_1-norm minimization. The selected genes have the intuitive interpretation of being the transcription factors regulating the target gene. The regression formulation is also highly parallelizable, as the resulting regression problems can be solved independently for each target gene.

However, existing regression based methods have a major limitation, namely: The regulation mechanism between every transcription factor-gene pair is assumed to be linear, meaning that the gene expression of the gene is modeled as as a linear function of the gene expression of the transcription factor. This assumption has a secondary consequence as well: *all* transcription regulations are assumed to be of the *same* type. Currently there is no simple way to accommodate multiple types of regulation functions in the regression class of algorithms. In contrast, biological networks are known to exhibit a multiplicity of regulation mechanisms including non-linear interactions. The periodic nature of the expression of the genes involved in cell-cycle regulation is a canonical example of nonlinear transcription regulation. Consider an example from the yeast cell-cycle, wherein expressions of the gene CLN3 and its transcription factor SWI5 oscillate together with some time lag [9]. For illustrative purposes, the expression levels of SWI5 and its target CLN3 can be modeled as functions of time by $x = \sin t$ and $y = \cos t$ respectively. Then a simple calculation shows that target gene expression is a nonlinear function of the expression of its transcription factor: $y = \sqrt{1 - x^2}$. Algorithms belonging to other categories such as mutual information or ensemble methods are able to accommodate different kind of transcription regulations including nonlinear functions. However, these algorithms do not yield simple and intuitive interpretations of the transcription regulation, as offered by the regression class of methods. This situation has motivated us to develop a new regression based method that circumvents the limitation of all transcription regulations being forced to be the same for all genes. Instead, a master list of possible regulation mechanisms is chosen *a priori*, and each transcription factor-gene pair is forced to choose one interaction from this list; however, the interactions could be different for different pairs. Thus our method allows for the multiplicity of regulation mechanisms capturing more biologically realistic situation, and also inherits the intuitive interpretation of the transcription regulation from the regression based methods. To validate this approach, we

• *The authors are with the Erik Jonsson School of Engineering and Computer Science, University of Texas at Dallas, Richardson, TX 75080. E-mail: {nitin.singh, m.vidyasagar}@utdallas.edu.*

Manuscript received 10 Dec. 2014; revised 20 June 2015; accepted 23 June 2015. Date of publication 29 June 2015; date of current version 30 Mar. 2016. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TCBB.2015.2450740

have benchmarked our method against the current state-of-the-art algorithms on the benchmark DREAM5 datasets [10]. Our method shows significant performance improvement over the winners of the DREAM5 competition.

The core of the proposed method relies on a modification to the least angle regression (LARS) model selection algorithm [11]. The acronym bLARS stands for block-wise least angle regression. The main idea behind the algorithm is briefly explained here, and complete details are given in Section 2.3. Suppose there are n candidate regulators for a given target gene, and denote their respective gene expressions by $\mathbf{a}_1, \ldots, \mathbf{a}_n$ for the candidate regulators and $\mathbf{y}$ for the regulated gene. The basic assumption is that the expression of the target gene can be approximated by a linear combination of appropriate functions of the expression levels of the candidate regulator genes. If there are k possible functions associated with each target regulator gene, then a direct solution would involve solving k^n linear regression problems. Clearly this is not computationally feasible. In the existing literature, this situation is tackled by further assuming that there is only one possible regulatory function for all genes; usually this function is taken to be linear. Then a single regression problem is solved for each target gene. The proposed method presents a strategy to circumvent this limitation, while at the same time avoiding the combinatorial explosion of having to solve k^n linear regression problems. The idea is to enforce a constraint that for each candidate regulator, only one of the possible regulatory functions is chosen, while permitting different candidate regulators to have different regulatory functions. This constraint is enforced via a simple extension to the LARS algorithm. Our experiments with the DREAM5 datasets show that the proposed greedy strategy for regulator selection indeed achieves higher overall score than any other algorithm.

Moreover, the regression formulation of GRN inference is very similar to feature-selection problems that are frequently encountered in the machine learning domain. Therefore, the presented method may have potential applications in solving general feature selection problems in other areas.

1.1 Literature Review

A GRN can be conveniently represented as a weighted directed graph, where each node represents a gene, and an edge from node i to node j means that gene i regulates gene j. The edge weights could correspond either to the strength of the regulation, or to the confidence we have that the regulatory mechanism really exists. In the present setting, by regulation we mean transcription regulation wherein the transcription factor physically binds to its target genes to activate or suppress their expression. However, genes can also influence each other via various other means such as proteins, metabolites and non-coding RNAs, to quote just a few possibilities. These broader types of regulation are not considered in the paper.

The early efforts to infer interactions amongst a set of genes from high-throughput data are based on clustering. The rationale is that genes with similar expression pattern should be functionally related [12]. These approaches use metrics such as the correlation coefficient or the Euclidean distance between the expression vectors of a pair of genes as a measure of their similarity. In these methods, only those edges whose correlation coefficients are more than a specified threshold value are retained. The resulting network is referred to as a "co-expression" network. The weighted gene co-expression network analysis (WGCNA) [13], [14] advocates using soft-thresholding with sigmoid or power functions to assign each edge a weight between 0 and 1. Since the WCGNA method is also based on the Pearson's correlation coefficient, it is unable to measure nonlinear relationships between pairs of genes. Another drawback of the WCGNA method is that it yields a complete graph where every node is connected to every other node; therefore, one has to threshold the edge weights in order to obtain the neighborhood of a gene, at which point WCGNA reduces to the co-expression network algorithm. In subsequent work, an alternative statistical measure of dependence, namely mutual information, is employed to detect dependence between a pair of genes [15]. In this paper, the mutual information between every pair of genes is calculated and is assigned as the weight of the corresponding edge. Then edges whose weights are less than a certain threshold are dropped, yielding a so-called Relevance Network. The main advantage of using mutual information over the correlation coefficient is that, whereas the correlation coefficient is invariant under any *affine* transformation of the expression values, mutual information is invariant under any *monotone* transformation of the data, which could also be nonlinear. However, neither mutual information based methods nor correlation based methods make any distinction between direct and indirect regulation. In other words, if gene i regulates gene j, and gene j regulates gene k, then the mutual information between genes i and k would also be large, even if there is no direct regulation.

Around the same time, graphical model based Bayesian methods have also been proposed [5], [16]. In these methods, a GRN is modelled as a directed acyclic graph (DAG). The DAG assumption permits the decomposition of the joint probability distribution of all the variables as a product of simpler joint probability distributions. However, biological networks are known to have cycles violating the Bayesian methods' acyclicity assumption.

The shortcomings of these methods led to the development of the next generation of methods such as CLR [2] and ARACNE [3] that allow cycles in the inferred GRN, and also make a distinction between direct and indirect regulation. Both of these methods use mutual information as the dependence measure between a pair of genes by treating their steady state expressions as random variables. However, instead of taking the raw mutual information score between a pair of genes, CLR computes a normalized z-score between every gene pair under the background distribution of mutual information scores of all of its neighbors. This step is performed to keep only the edges that show higher than the background distribution of the mutual information in the vicinity. On the other hand, ARACNE employs the data processing inequality [17, p. 34] to prune the edges of the GRN by considering all possible triplets of genes. Both of these methods build upon [15] and have been proved to be quite popular. Though these methods allow the GRN to contain cycles, the resulting GRNs are undirected graphs, because the mutual information is a symmetric quantity. This means that the presence of an

edge between a pair of genes does not have any indication of the direction. In [3], the pairs of genes studied consist of one known transcription factor and another gene; therefore the direction of the regulation, if it exists, can be inferred. However, in the cases when the objective is to construct *genome-wide* regulatory networks where all of the transcription factors are not known, the inability to infer the direction of the regulation is a major limitation.

The application of regression techniques such as LASSO [18] has led to development of several GRN inference methods that also allow edges to have direction [4], [19]. In general, the regression based methods make the assumption that every gene is regulated by a comparatively small number of regulators. This biologically meaningful sparsity assumption is then enforced by l_1-norm minimization techniques to select a small number of potential regulators from a possibly large list of candidate regulators. This approach is analogous to solving under-determined feature selection problems, which is the original motivation for the invention of the LASSO algorithm. In a typical problem instance, the number of available samples is much smaller then the number of regressor variables (genes). To handle this under-determined problem, these methods resort methods such as ℓ_1-norm minimization for feature selection, and to some model averaging or bootstrapping method to achieve robustness. A tree-based ensemble method, GENIE3 [7] was shown to outperform several other methods [1]. There are other tree-based ensemble methods such as ADANET [20] that employ gradient boosting of regression trees. The ANOVA based method [8] performs well in some cases but depends on additional perturbation data. In addition, there exist meta-approaches that combine two or more of the GRN inference methods to get a better prediction [21]. In fact, a recent paper [1] argues that combining multiple approaches yields a better GRN, and terms this phenomenon as "the wisdom of crowds." The reader is referred to [1], [22], [23] and references therein for a detailed review.

The methods reviewed thus far rely on steady-state gene expression data. A variety of methods such as Bayesian Model Averaging [24] or ODE based modeling [24], [25], [26] are employed for time series gene expression data. In addition to the described approaches that rely on gene expression values, there exists a body of work that uses sequence alignment to search for transcription factor binding sites in the promoter region of a gene to determine the regulation. Some papers that pursue this approach are [27], [28].

2 The bLARS Approach

2.1 Problem Formulation

It is assumed that the expression value of a gene is a weighted linear sum of possibly nonlinear functions of the expression levels of its regulators. Formally, let $\mathbf{y} \in \mathbb{R}^{m\times 1}$ denote the vector of expression levels of a gene in m different samples, and let $A \in \mathbb{R}^{m\times n}$ denote the matrix of gene expression levels of n candidate regulator genes in the same m samples. Then as per our assumption, we can write

$$\mathbf{y} = [\boldsymbol{\psi}_1(\mathbf{a}_1) \dots \boldsymbol{\psi}_n(\mathbf{a}_n)]\mathbf{w}, \tag{1}$$

where $\mathbf{a}_i$ denotes the ith column of the matrix A. Further, $\boldsymbol{\psi}_i(\cdot)$ is a nonlinear basis function that captures the manner in which the ith candidate regulator gene regulates the target gene. These functions are assumed to be continuous. The weight vector $\mathbf{w} = [w_1, w_2, \dots, w_n]^T$ denotes the strength of these regulations. So the assumption is that if w_j is zero then gene j does not regulate the target gene. This is a generalization of the formulations presented in previous work; see e.g. [7]. Typically, within the regression class of GRN inference algorithms, (1) is solved using standard linear regression techniques such as LASSO [18], so as to yield a sparse weight vector $\mathbf{w}$. The sparsity assumption is based on the biological observation that any gene has only a handful of regulators from possibly very large number of potential regulators. To get the complete GRN, (1) is solved for each gene in turn by treating the remaining genes as potential regulators.

In order to simplify the linear regression problem, most algorithms typically make the assumption that all of the regulation functions $\boldsymbol{\psi}_i(\cdot)$ are the same; most often they are all chosen to be linear. In biological networks, however, different regulators can regulate the target gene in different ways. But simply allowing multiple options for the regulation function of each candidate regulator gene is computationally infeasible, because this would result in an exponential number of instances of equations (1). If there are k choices for each $\boldsymbol{\psi}_i(\cdot)$, there will be k^n versions of (1) to be solved. In this work, we present a strategy to solve (1) only once, but still permit each regulator to choose from k possible regulation functions. These possible regulation functions need to be specified *a priori*.

Let us consider the general model presented in (1) and assume that there are k possible choices for the basis function $\psi_i(\cdot)$. Given these possible k choices, say $f_1, f_2, \dots, f_k$, we apply these functions to every column $\mathbf{a}_i$ of A. This leads to the following formulation:

$$\begin{aligned}\mathbf{y} &= [\underbrace{f_1(\mathbf{a}_1), f_2(\mathbf{a}_1), \dots, f_k(\mathbf{a}_1)}_{b_1}, \dots, \underbrace{f_1(\mathbf{a}_n), f_2(\mathbf{a}_n), \dots, f_k(\mathbf{a}_n)}_{b_n}]\boldsymbol{\beta} \\ &= X\boldsymbol{\beta},\end{aligned} \tag{2}$$

where $X \in \mathbb{R}^{m\times nk}$, $\beta \in \mathbb{R}^{nk\times 1}$, and each block b_j is of size k representing all possible ways in which gene j can regulate the target. Similarly, one can also partition $\boldsymbol{\beta}$ commensurately into n equal partitions of size k. This formulation allows a regulator to choose a function from a predefined collection of k functions. Now we wish to select *at most* one column from any block b_j corresponding to the function that best captures the regulation mechanism. In other words, we seek a sparse solution of (2) such that *at most* one entry of any block b_j of $\boldsymbol{\beta}$ is nonzero. If all entries of a block b_j are zero, then the jth candidate regulator does not in fact regulate the target gene.

This requirement is quite different from what existing group based feature selection techniques such as Group LASSO [29] attempt to achieve. Specifically, the Group LASSO algorithm selects or rejects an *entire* block (group) of features by adding the block-wise l_2-norm of the vector $\boldsymbol{\beta}$ as an regularization term to the objective function. However, the problem formulation (2) requires the algorithm to select *at most* one feature from a block instead of selecting *all* the features from a block. Therefore the Group LASSO can not

be used to solve this problem. A variant of the Group LASSO algorithm, namely the Sparse Group Lasso algorithm [30], aims to promote both the selection of a small number of blocks, as well as sparsity within the selected blocks (groups) of variables. The Sparse Group Lasso algorithm uses a convex combination of the group wise l_2-norm of the $\boldsymbol{\beta}$ and l_1-norm of the $\boldsymbol{\beta}$ vector. However, while the Sparse Group Lasso algorithm can promote "within group sparsity" by controlling the regularization parameter, it can not guarantee the selection of *at most* one feature per group, which is the requirement of the present problem formulation.

In order to achieve this objective, we propose a modification of the least angle regression [11] model selection technique. There are several advantages of choosing the LARS algorithm for the problem of GRN inference. First, it is possible to modify the LARS algorithm to yield the required block sparsity structure in the $\boldsymbol{\beta}$ vector. Second, LARS algorithm has low computational complexity i.e. equal to the ordinary least-square algorithm. Third, it is numerically efficient when the number of covariates is greater than the number of samples, a situation typically encountered in the GRN inference problem. These properties of the LARS algorithm makes it very suitable to our purpose. However, we note that the LARS algorithm could be susceptible to noise in the input data. The greedy step wise nature of the LARS algorithm adds a new feature to the model at every successive step. In the context of our problem, at every consecutive step LARS selects a regulator that reduces the residual (error) of the model. Next we first describe LARS algorithm briefly and then present our proposed modification.

2.2 The LARS Agorithm

LARS builds upon two regression methods, namely Forward Stepwise Regression and Forward Stagewise Regression. These classical model selection techniques first choose a set of predictors from a given collection of predictors, and then build a linear model based on the selected set. LARS is a "stylized" version of the Forward Stagewise Regression method. By exploiting the geometry of the covariate selection procedure, LARS derives a formula that allows the Forward Stagewise Regression method to take much larger steps; however LARS remains less greedy than the Forward Stepwise Regression algorithm described below.

Let us suppose without loss of generality that the response and covariates are normalized so as to have zero mean and unit length, that is, $\sum_{i=1}^{m} y_i = 0$, and $\sum_{i=1}^{m} X_{ij} = 0, \|X_j\|^2 = 1, \text{ for } j = 1, 2, \ldots, nk$ where $\|\cdot\|$ denotes the l_2 norm of a vector. Let $\hat{\boldsymbol{\beta}}$ denote the current regression coefficient, and define the corresponding prediction vector $\hat{\boldsymbol{\mu}} := X\hat{\boldsymbol{\beta}}$. Also, let $\hat{\mathbf{c}}$ denote the vector of the *current correlations* of the *residual* with the covariates, that is

$$\hat{\mathbf{c}} = \mathbf{c}(\hat{\boldsymbol{\mu}}) = X^T(\mathbf{y} - \hat{\boldsymbol{\mu}}) \tag{3}$$

such that $\hat{c}_j = X_j^T(\mathbf{y} - \hat{\boldsymbol{\mu}})$ is proportional to the correlation coefficient of the jth covariate with the current *residual*. Starting from the zero prediction vector $\hat{\boldsymbol{\mu}} = 0$, the two classic algorithms proceed in the direction of the covariate that has highest *current correlation* in the following iterative fashion:

$$i = \underset{j}{\operatorname{argmax}} |\hat{c}_j| \text{ and update } \hat{\boldsymbol{\mu}} \leftarrow \hat{\boldsymbol{\mu}} + \gamma_i \cdot \operatorname{sign}(\hat{c}_i) \cdot X_i \tag{4}$$

with step sizes γ_i. The value of γ_i is set to a *small* fixed value ϵ in Stagewise Regression and to $|\hat{c}_i|$ in Stepwise Regression. Thus Forward Stepwise Regression takes an overly greedy stance, selecting the predictor X_i that has the highest current absolute correlation, while discarding all other predictors that are highly correlated to X_i. On the other hand, Forward Stagewise Regression proceeds in a much more conservative fashion by taking very large number of small ϵ-sized steps while selecting a covariate predictor. These large number of substeps make Forward Stagewise Regression a time intensive algorithm. LARS essentially follows a covariate selection path that is similar to the Forward Stagewise approach, but uses an accelerated computation strategy that allows the algorithm to add a new covariate to the model at every step of iteration.

Starting from the zero prediction estimate $\hat{\boldsymbol{\mu}} = 0$, at every step, LARS successively adds one covariate to the current set of selected covariates, referred to as the *active set*. However, in contrast with the Stagewise and Stepwise methods, LARS moves in a direction that is *equiangular* to all covariates in the *active* set until a new covariate becomes equally correlated with the residual. At this point, the new covariate is added to the *active* set, causing a change in the LARS direction. Suppose $\mathcal{A} \subset \{1, 2, \ldots, nk\}$ denotes the set of currently active indices. It is assumed that the covariates indexed by $\mathcal{A}$ are linearly independent. Then in the next iteration, the LARS algorithm updates the current prediction estimate $\hat{\boldsymbol{\mu}}_{\mathcal{A}}$ to

$$\hat{\boldsymbol{\mu}}_{\mathcal{A}_+} = \hat{\boldsymbol{\mu}}_{\mathcal{A}} + \hat{\gamma}\mathbf{u}_{\mathcal{A}}, \tag{5}$$

where $\hat{\gamma}$ is the step size taken in the direction of unit vector $\mathbf{u}_{\mathcal{A}}$ that is *equiangular* to active set covariates. The key idea behind the LARS algorithm is that there exists a closed-form formula for computing the step size $\hat{\gamma}$ that makes a new covariate to be equally correlated with the current residual. With the current estimate $\hat{\boldsymbol{\mu}}_{\mathcal{A}}$,

$$\hat{\mathbf{c}} = \mathbf{c}(\hat{\boldsymbol{\mu}}_{\mathcal{A}}) = X^T(\mathbf{y} - \hat{\boldsymbol{\mu}}_{\mathcal{A}}) \tag{6}$$

is defined as the vector of current correlations. Furthermore, denote the maximum absolute correlation by $\hat{C} = \max_j (|\hat{c}_j|)$. Note that all active set columns have the greatest absolute current correlation, or in symbols, $|\hat{c}_j| = \hat{C}$, $j \in \mathcal{A}$. Let $s_j = \operatorname{sign}(\hat{c}_j), j \in \mathcal{A}$ and define

$$X_{\mathcal{A}} = (s_j X_j), \quad j \in \mathcal{A}, \tag{7}$$

a submatrix of X whose columns are indexed by the active set $\mathcal{A}$ and multiplied by their corresponding correlation sign with the residual. Now let

$$\mathcal{G}_A = X_{\mathcal{A}}^T X_{\mathcal{A}} \text{ and } A_{\mathcal{A}} = (1_{\mathcal{A}}^T \mathcal{G}_{\mathcal{A}} 1_{\mathcal{A}})^{-1/2}, \tag{8}$$

where $1_{\mathcal{A}}$ is a vector of all ones with length $|\mathcal{A}|$ and $A_{\mathcal{A}}$ is a number that can be thought as a normalization factor. Then the vector

$$\mathbf{u}_{\mathcal{A}} = X_{\mathcal{A}}(A_{\mathcal{A}} \mathcal{G}_{\mathcal{A}}^{-1} 1_{\mathcal{A}}), \tag{9}$$

makes an equal and acute angle from all active set columns and is of unit length. This can be seen as follows:

$$\begin{aligned}\|\mathbf{u}_{\mathcal{A}}\|^2 = \mathbf{u}_{\mathcal{A}}^T\mathbf{u}_{\mathcal{A}} &= \left(X_{\mathcal{A}}(A_{\mathcal{A}}\mathcal{G}_{\mathcal{A}}^{-1}1_{\mathcal{A}})\right)^T\left(X_{\mathcal{A}}(A_{\mathcal{A}}\mathcal{G}_{\mathcal{A}}^{-1}1_{\mathcal{A}})\right)\\ &= (A_{\mathcal{A}}\mathcal{G}_{\mathcal{A}}^{-1}1_{\mathcal{A}})^T X_{\mathcal{A}}^T X_{\mathcal{A}}(A_{\mathcal{A}}\mathcal{G}_{\mathcal{A}}^{-1}1_{\mathcal{A}})\\ &= A_{\mathcal{A}}^2(\mathcal{G}_{\mathcal{A}}^{-1}1_{\mathcal{A}})^T X_{\mathcal{A}}^T X_{\mathcal{A}}(\mathcal{G}_{\mathcal{A}}^{-1}1_{\mathcal{A}})\\ &= A_{\mathcal{A}}^2(1_{\mathcal{A}}^T\mathcal{G}_{\mathcal{A}}^{-1}1_{\mathcal{A}}) = 1,\end{aligned} \tag{10}$$

where the first step follows from (9) and the last two steps use (8) and the fact that $\mathcal{G}_{\mathcal{A}}$ is symmetric matrix. Furthermore, a simple calculation show that the angle between $\mathbf{u}_{\mathcal{A}}$ and the covariates of the active set is indeed equal, because

$$\begin{aligned}X_{\mathcal{A}}^T\mathbf{u}_{\mathcal{A}} &= X^T X_{\mathcal{A}}(A_{\mathcal{A}}\mathcal{G}_{\mathcal{A}}^{-1}1_{\mathcal{A}})\\ &= A_{\mathcal{A}}(X^T X_{\mathcal{A}}\mathcal{G}_{\mathcal{A}}^{-1})1_{\mathcal{A}}\\ &= A_{\mathcal{A}}1_{\mathcal{A}}.\end{aligned} \tag{11}$$

Once equiangular vector $\mathbf{u}_{\mathcal{A}}$ is computed, the only thing remaining is to calculate is the step size $\hat{\gamma}$. Let us define

$$\mathbf{a} := X^T\mathbf{u}_A, \tag{12}$$

a vector that measures the angle between the $\mathbf{u}_{\mathcal{A}}$ and *all* covariates. Also define

$$\boldsymbol{\mu}(\gamma) = \hat{\boldsymbol{\mu}}_{\mathcal{A}} + \gamma\mathbf{u}_{\mathcal{A}} \quad \forall\gamma > 0. \tag{13}$$

Then the current correlation can be computed using by (6) and (12) as

$$\begin{aligned}c_j(\gamma) &= X_j^T(\mathbf{y} - \boldsymbol{\mu}(\gamma))\\ &= X_j^T(\mathbf{y} - \boldsymbol{\mu}_{\mathcal{A}}) - \gamma X_j^T\mathbf{u}_{\mathcal{A}}\\ &= c_j - \gamma a_j.\end{aligned} \tag{14}$$

Similarly for any active set column $j \in \mathcal{A}$, the updated correlation with the residual yields

$$|c_j(\gamma)| = \hat{C} - \gamma A_{\mathcal{A}} \quad \text{using (11) as } j \in \mathcal{A}. \tag{15}$$

Now by equating (15) and (14), we arrive at the conclusion that for $j \in \mathcal{A}^c$, $c_j(\gamma)$ attains maximal value for $\gamma = (\hat{C} - \hat{c}_j)/(A_{\mathcal{A}} - a_j)$ or $\gamma = (\hat{C} + \hat{c}_j)/(A_{\mathcal{A}} + a_j)$. Thus the minimum positive step size is

$$\hat{\gamma} = \min_{j\in\mathcal{A}^c}\left\{\frac{\hat{C} - \hat{c}_j}{A_{\mathcal{A}} - a_j}, \frac{\hat{C} + \hat{c}_j}{A_{\mathcal{A}} + a_j}\right\}_+ \tag{16}$$

such that a new index, say $\hat{j} \in \mathcal{A}^c$, which minimizes (16), enters the active set. Therefore the updated active set $\mathcal{A}_+ = \mathcal{A} \cup \{\hat{j}\}$ expands in every successive step which is akin to adding a new covariate to the model. The updated absolute correlation is $|c_j(\gamma)| = \hat{C} - \hat{\gamma}A_{\mathcal{A}}$ that means that the correlation of all active set columns reduces by an equal amount. Note that in addition to providing the selected covariates, the LARS algorithm also presents the covariate selection path, an ordered list of columns, which is utilized by the downstream scoring method in the bLARS algorithm. The reader is referred to [11] for a detailed description of the LARS algorithm.

2.3 Proposed Modification to the LARS Algorithm

In order to enforce the block sparsity structure on the chosen vector $\boldsymbol{\beta}$, we propose the following modification of LARS. Whenever a column from block b_j of X is chosen in the active set $\mathcal{A}$, the remaining columns of that block are excluded from all future computations of the successive active coavariate in (16). As a result, at most one column from any block is selected, leading to the desired block sparsity structure of $\boldsymbol{\beta}$. If no column from the block b_j is selected, then the jth regulator does not regulate the target gene. Now the weight vector $\mathbf{w}$ in (1) is obtained by performing a block-wise OR operation on $\boldsymbol{\beta}$. This means that if any entry of the block b_j is non-zero, then that value is assigned to $\mathbf{w}_j$, and the remaining components of $\mathbf{w}$ are set to zero. Thus after L-steps, an ordered list of columns corresponding to non-zero weights are generated.

3 The bLARS Algorithm

Fig. 1 shows an illustration of the bLARS algorithm. First, a gene is designated as a target gene and then its regulators are predicted for several bootstrap runs. A scoring method is employed to aggregate the predictions over all bootstrap runs yielding a score for each candidate regulator. It is worth noting that the bootstrap runs for each target gene are independent and can thus be carried out in parallel. The overall GRN is a collection of these individual target genes and their selected regulators. A description of individual steps of the algorithm is given next.

3.1 Bootstrapping

The bLARS algorithm uses bootstrapping in order to obtain a more reliable selection of the regulators. In general, the bootstrap procedure [31] is used to estimate the parameters from the empirical distribution. In simple terms, bootstrapping means generating multiple sets of samples from the observed samples by *resampling*, and then computing the parameter of interest for each *resampled* set. Finally, an estimate of the parameter in question is obtained by averaging over all of the resampled sets. In *resampling*, samples are randomly selected (uniformly at random, with replacement) from the observed samples. The bootstrapping technique is frequently applied to get stable results in the case of underdetermined problems [32]. In the current bLARS implementation, the *resampling* is performed $r = 500$ times. In each of these bootstrap runs, $\mathbf{y}$ and A are chosen uniformly at random with replacement from the given gene-expression data. Subsequently, the LARS with the proposed modification is utilized to select the regulators for each of these bootstraps. Finally, the results of all bootstrap runs are aggregated using an area-based scoring [4] technique that is described later in this section. Note that the bLARS algorithm employs bootstrapping *only* to select the high-confidence regulators for each target gene, and it does not aggregate over many bootstrap networks which is a characteristic of the ensemble methods. The selected genes still maintain the intuition of being regulators of the target gene.

3.2 Modified LARS Based Regulator Selection

The key differentiator of the bLARS algorithm is that it permits different regulator functions for different candidate

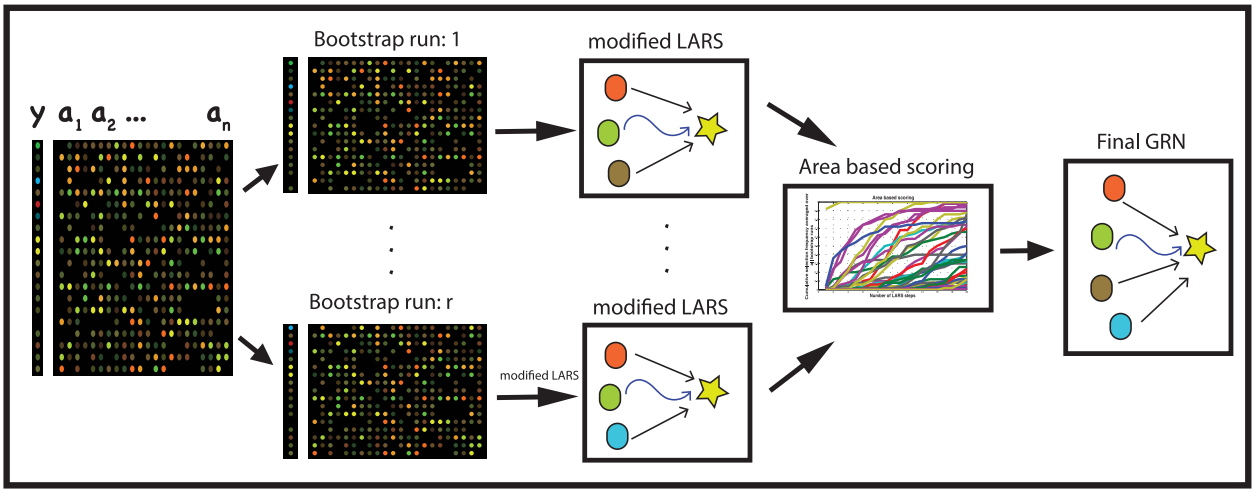

Fig. 1. A flow chart of the bLARS algorithm. First, a gene is designated as target (y) and rest are treated as potential regulators ($\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n$). Then, the steady state gene-expression is *resampled* for r bootstrap runs. For each of these runs, modified bLARS algorithm is utilized to select the regulators from all the candidates. Finally, an area based method is used to aggregate the prediction over all bootstrap runs to produce the final GRN for the target gene. These steps are repeated for each gene in turn to get the overall GRN.

regulators. The possible regulation mechanisms are captured by basis functions $f_1, f_2, \ldots, f_k$ described in Equation (2). The current implementation of bLARS contains five basis functions, namely: linear, logistic up-regulation, logistic down regulation, cubic and square root. The proposed LARS modification (Section 2.3) selects L possible regulators from the set of all possible candidate regulators after L steps. Thus the parameter L represents the average number of transcription factors for the given organism. In the algorithm, the number of LARS steps i.e. L was set to 25. The number and type of basis functions as well as value of the parameter L can be specified by the user.

3.3 Area-Based Scoring

The objective of the area-based scoring method is to assign a score to each candidate regulator corresponding to the frequency of its selection over several bootstrap runs. This scoring method exploits the ordering information about the selection of the regulators that is provided by the LARS algorithm. This is achieved via area based scoring method as follows.

Let $\phi_{il}, i = 1, \ldots, n \quad l = 1, \ldots, L$ be cumulative selection frequency of ith regulator in the lth LARS step. The average here was taken over all bootstrap runs. Then the score for each regulator is defined as

$$s_i = \frac{1}{L} \sum_{l=1}^{L} \phi_{il},$$

which is simply the sum of cumulative selection frequency averaged over L LARS steps. For illustration, a simple toy example is shown in Fig. 2. The example shows the

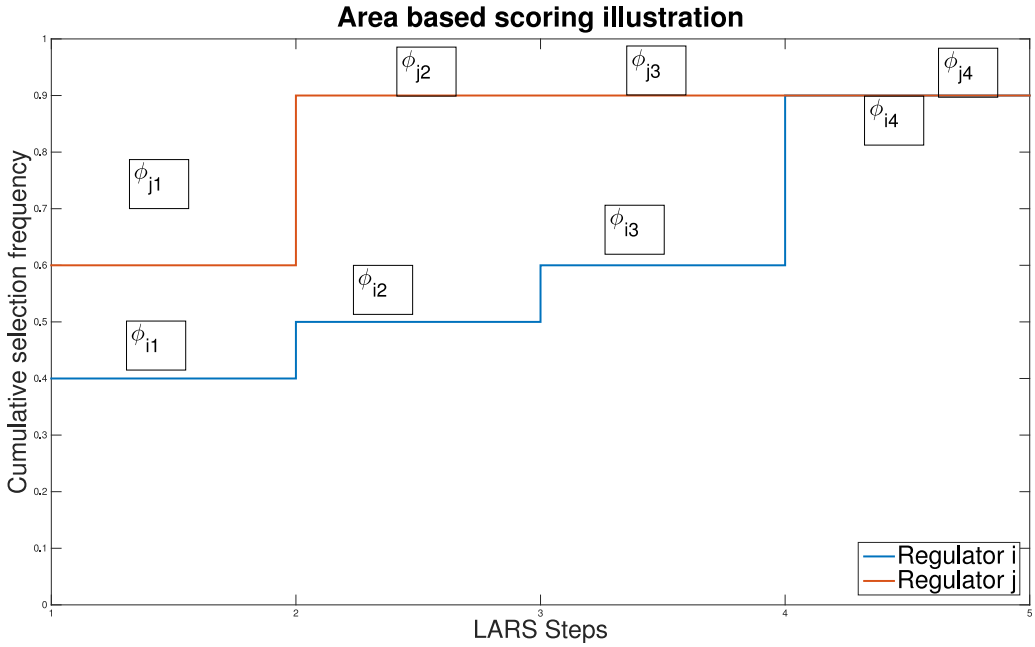

Fig. 2. An illustration of the area based scoring method. The cumulative selection frequencies of only two regulators i and j are shown. The jth regulator achieves higher score than the ith regulator, that is $s_j > s_i$, even though both of them have same overall cumulative selection frequency of 0.9.

cumulative selection frequencies of two regulators over five LARS steps. The values $\phi_{i1} = 0.4, \phi_{i2} = 0.5$ mean that, over all bootstrap runs, the ith regulator was selected 40 percent of the time in the first LARS step and 10 percent of the time in the second LARS step. That is why the cumulative selection frequency ϕ_{i2} is 50 percent. Now the score s_i has a natural interpretation of the area under the cumulative selection frequency curve normalized by the total area L. The score s_i is assigned as the edge weight of the regulator i and the target gene. Clearly this score not only takes into account the overall selection frequency of a transcription factor but also rewards the selection in the earlier LARS steps. This method is less sensitive to the number of LARS steps than simple ranking based on overall selection frequency ϕ_{iL}. In earlier work [4], area based scoring was used in the context of stability selection [33]; however, its application to bootstrapping is new.

3.4 bLARS Implementation

The pseudo-code for the bLARS algorithm is given below. Here A_j refers to the jth column of the matrix A, and A_I is the submatrix that contains only the columns of A that are members of the index set I. Suppose that the input data matrix $\mathbf{A} \in \mathbb{R}^{m \times n'}$ and the indices of the transcription factors I are given. Then for each target gene i, and bootstrap run j, the respective values of $\mathbf{y}$ and A are obtained by *resampling* from the expression of the target gene $\mathbf{A}_i$ and the expression values of the remaining transcription factors $\mathbf{A}_{\{I \setminus i\}}$. At this stage, the modified LARS algorithm is invoked to return an ordered list of selected transcription factors denoted by S_j. Finally, after all bootstrap runs, the matrix S is passed to the area based scoring method that assigns a score between 0 and 1 to each transcription factor to the target gene edge.

Algorithm 1. bLARS Pseudo Code

Input: $\mathbf{A} \in \mathbb{R}^{m \times n'}, I \subset \{1, \ldots, n'\}, |I| = n$ ▷ m samples, n' genes, Index of n tx-factors
Set r, L ▷ Number of bootstraps and LARS steps
Initialize $W \in \mathbb{R}^{n \times n'}$ ▷ Adjacency matrix of the GRN
for $i \leftarrow 1, n'$ **do** ▷ For each target gene
  Initialize $S \in R^{n \times r}$ ▷ Initialize selection matrix
  for $j \leftarrow 1, r$ **do** ▷ For each bootstrap run
    $\mathbf{y} = resample(\mathbf{A}_i), A = resample(\mathbf{A}_{\{I \setminus i\}})$ ▷ *Resampling*
    $S_j =$ modified-blars$(\mathbf{y}, A, L)$ ▷Returns selected tx-factors
  end for
  $W_i =$ area-score(S, L, r) ▷ Area based scoring
end for
Output: W ▷ Adjacency matrix of the GRN

The time-complexity of the algorithm depends mainly on LARS which takes $O((nk)^3)$ for each target gene where n is number of transcription factors and k is the number of basis functions. Thus the overall time-complexity of the algorithm is $O(n^4k^3)$. A Matlab implementation of bLARS and bench mark DREAM5 datasets are made available at the web site *http://sourceforge.net/projects/blars*. As mentioned above, the computation is completely parallelizable and can be carried out simultaneously on several machines in a cluster. Also, implementing bLARS in a more efficient low level language such as C would reduce the CPU time even further.

4 Results

4.1 Input Data: DREAM5 Networks

The inferencing of GRNs has been quite an active area of research during the past decade. Consequently, a community based consortium called "Dialogue for Reverse Engineering Assessments and Methods" (DREAM) [34] has emerged. The DREAM consortium holds international reverse engineering challenges, providing standardized common input datasets and performance evaluation metrics to compare different approaches. The DREAM datasets have become a standard benchmark in the GRN inference community and are frequently used to assess the performance of new algorithms [4], [7], [35]. The latest DREAM5 challenge has released three gene-expression datasets and corresponding gold-standard networks, namely: *in silico*, *E. coli* and *S. cerevisiae*. In this paper, these networks are also referred to as Network 1, Network 2, and Network 3 respectively. The topology of the synthetic (*in silico*) network is based on the sub-networks of the model organisms which accounts for the network features such as the feedback loops and indirect interactions. The gene expression simulation is based on detailed kinetic model of the gene-regulation, capturing both independent and synergistic interactions, transcription and translation dynamics, and stochastic noise. An additional source of noise based on the observed noise models in the microarray data is added to the synthetic data generation process [10]. The simulation is done with the GeneNetWeaver [36] tool. The DREAM consortium has not revealed *in silico* simulation settings, and anonymized the E. coli and Yeast (*S. cerevisiae*) datasets.

The input gene expression and gold-standard networks for these organisms were obtained from DREAM5 website [10]. The *in silico* data contains simulated gene-expression values of 805 samples for 1,843 genes. Out of these 1,843 genes, 195 genes are designated as transcription factors, that is, potential regulators. The E. coli dataset has gene-expression values of 805 samples for 4,511 genes including 334 designated transcription factors. The Yeast dataset has expression level of 536 samples for 5,950 genes that includes 333 transcription factors. In all of the inferred networks, only the transcription factors were treated as potential regulators for each target gene. The gold standard GRNs for *in silico*, E. coli and Yeast have 4,012, 2,066 and 3,940 edges respectively. Note that the E. coli gold-standard GRN has fewer edges than the other two networks. In addition, the DREAM dataset provides the empirical null distribution of AUROC and AUPR scores simulated over 25,000 randomly generated networks. The p-value of the predicted GRN is computed with respect to these null distributions. A detailed description of the input datasets, gold standard networks and the null distributions can be found in the supplementary information, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TCBB.2015.2450740, of the DREAM5 paper [1].

4.2 Performance Evaluation Metrics

To evaluate the results of the GRN inference algorithms, we used the standard performance metrics recommended by the DREAM challenge, namely the Area under receiver operating characteristic (AUROC), and the area under

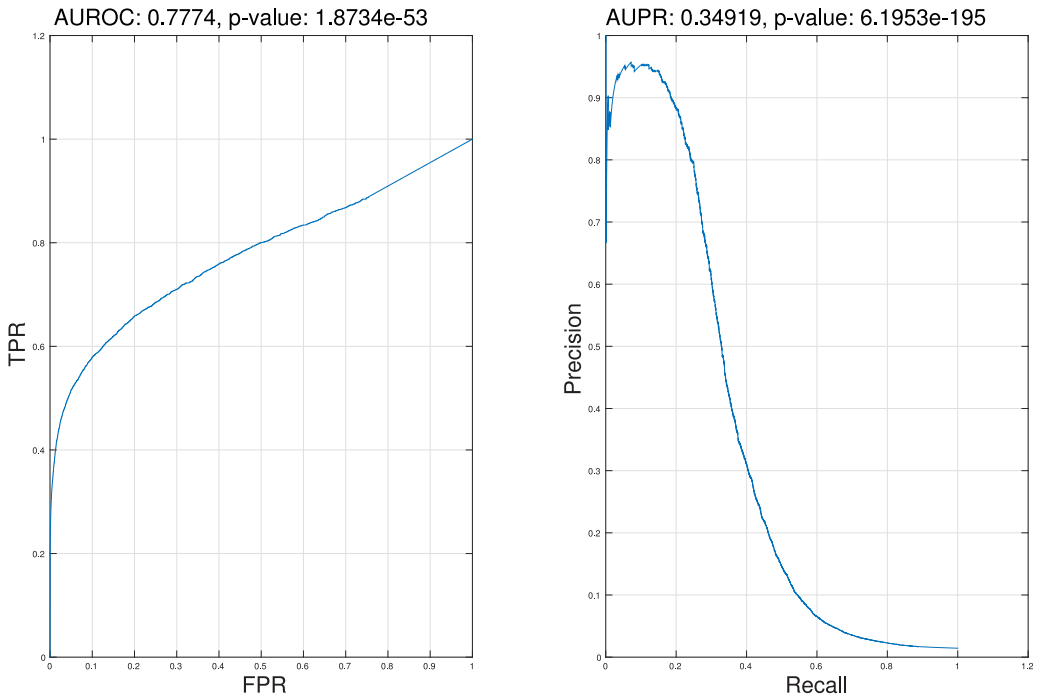

Fig. 3. ROC and PR curves for the Network 1. The respective AUROC and AUPR and p-values are shown in the title. The network score of the Network 1 can be calculated by (21) i.e. $\text{NW Score}_1 = -0.5\log_{10}(1.87 \cdot 10^{-53} \times 6.19 \cdot 10^{-195}) = 123.46$.

precision recall (AUPR). Using these metrics, the predicted network edges are compared against the gold-standard network edges, treating it as a binary classification task. To calculate both of these metrics, predicted GRN edges are first sorted by their weights; then the true-positive (TP) and false-positive (FP) *rates* are defined as functions of the top k edges in the sorted edge-list of predictions, as follows:

$$\text{TPR}(k) := \frac{\text{TP}(k)}{\text{P}}, \tag{17}$$

$$\text{FPR}(k) := \frac{\text{FP}(k)}{\text{N}}, \tag{18}$$

$$\text{recall}(k) := \frac{\text{TP}(k)}{\text{P}}, \tag{19}$$

$$\text{precision}(k) := \frac{\text{TP}(k)}{k}, \tag{20}$$

where TP(k) is number of predicted edges among top k edges that are also present in gold-standard (GS) network; FP(k) is number of predicted edges in top k edges that are not in GS, while P and N are number of positive and negative (absent) edges in GS respectively. Now, one can plot the TPR(k) vs. FPR(k) and precision(k) vs. recall(k) rates for $k = 1, \ldots, K$, where K denotes the total number of edges predicted by the algorithm, to obtain ROC and PR curves respectively. In other words, each value of k generates a point on the PR and ROC curves. Intuitively, these rates corresponds to the sensitivity (precision) and specificity (recall) of the predictions considering only top k edge predictions for $k = 1, \ldots, K$ in the ROC (PR) curve. The areas under these curves are used as metrics to measure the performance of a GRN inference algorithm and is referred to as AUROC score and AUPR score.

Further, an empirical p-value is computed for each predicted network using as the background null distribution the AUPR and AUROC values of $25,000$ random networks generated in the DREAM5 challenge. To compare the performance of these algorithms on individual networks, one can define a network score as the average of (the decimal logarithm of) the p-values of AUROC and AUPR of the network wise predictions. This simply reflects how well an algorithm performs on individual networks

$$\text{NW Score}_i = -\frac{1}{2}\log_{10}(p\text{-AUROC}_i \times p\text{-AUPR}_i) \tag{21}$$

The Fig. 3 shows the ROC and PR curves, their respective AUROC and AUPR scores and p-values for the synthetic network i.e. Network 1 to illustrate the score computation procedure.

Next, the AUROC and AUPR scores are obtained by taking the average of the decimal logarithm of the p-values of all three networks, respectively

$$\text{AUROC Score} = -\frac{1}{3}\sum_{i=1}^{3}\log_{10}(p\text{-AUROC}_i), \tag{22}$$

$$\text{AUPR Score} = -\frac{1}{3}\sum_{i=1}^{3}\log_{10}(p\text{-AUPR}_i). \tag{23}$$

The AUPR and AUROC scores measure how well an algorithm performs across the network using AUPR or AUROC as the performance metric. Finally, the overall score is just the mean of these two scores. Intuitively, the overall score measures the overall performance of an algorithm across different networks averaged over both the AUPR and AUROC metrics.

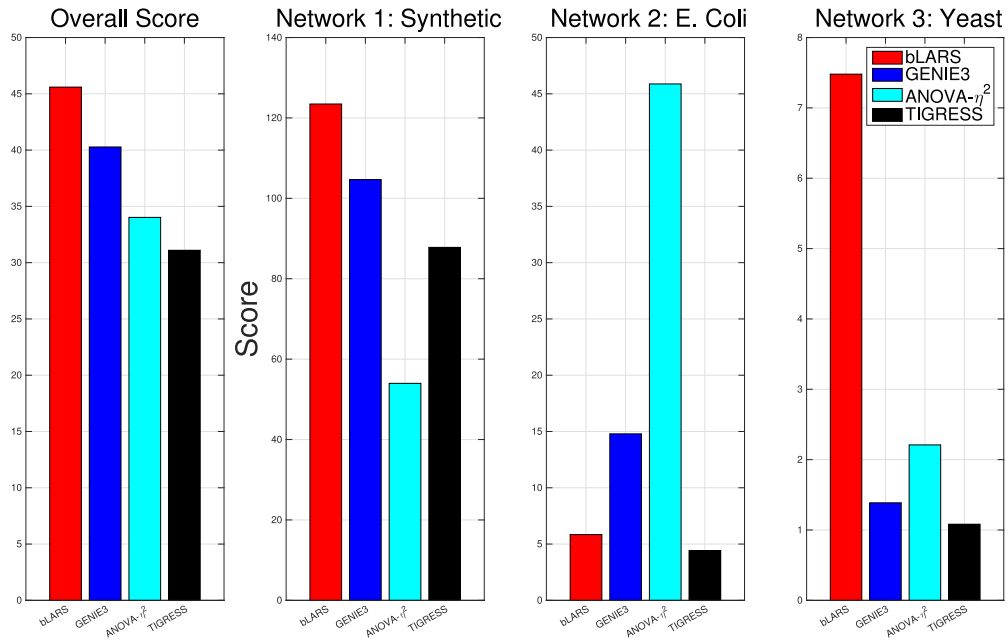

Fig. 4. Network-wise performance comparison of the bLARS algorithm with DREAM5 top performers. A higher score means better performance.The overall score is the average of all the three network wise scores.The bLARS achieves the highest overall score, and highest network wise scores for the synthetic and Yeast networks.

4.3 Performance of bLARS

The performance of the bLARS algorithm is first compared with the three top performing algorithms in the DREAM5 GRN inference competition, namely: GENIE3, ANOVA-η^2 and TIGRESS. The GENIE3 algorithm is based on ensemble of regression trees, TIGRESS is a regression based method with stability selection, and ANOVA-η^2 method utilizes η^2, a nonlinear correlation measure based on ANOVA. The overall scores as well as scores on individual networks are shown in Fig. 4. The bLARS algorithm achieves a higher overall score than these three state-of-the-art algorithms. The remaining three panels compare the performance of these algorithms on three individual datasets. Note that bLARS outperforms the other there algorithms on the *in silico* and Yeast networks, and ranks third in the case of E. coli GRN. The ANOVA-η^2 performs best on the E. Coli network. However, it is noteworthy that the ANOVA-η^2 algorithm takes additional meta-data into account to infer the GRN. Specifically, ANOVA-η^2 makes use of perturbation data such as gene-knockout or over-expression of specific transcription factors. This type of information requires additional elaborate experimentation, limiting the applicability of the algorithms that depends on the additional meta-data.

Next we expand the scope of the comparison, wherein the performance of the bLARS algorithm is compared with the all 29 algorithms that participated in the DREAM5 challenge. These algorithms can be divided into two categories:

1) The first category comprises the algorithms that either seek additional information generated by perturbation experiments, or combine multiple GRN inference algorithms. Application of the former type of algorithms is limited as the perturbation information needs additional experiments. Further, the latter type of "meta-algorithms" are distinct from novel individual methods. Indeed, a recent DREAM consortium paper [1] argues that a "community network," which is obtained by integrating results of the individual methods, is better than the *most of the individual methods*. This phenomenon is attributed to the complementary strengths of algorithms belonging to different classes. Therefore, these "meta-algorithms" are in a sense "community methods" albeit based on only handful number of *individual methods*. Examples of such algorithms are: ANOVA-η^2 [8], and methods annotated as META in Table 1.
2) The second category contains all the algorithms that does not belong to the first category. Thus this category essentially consists of the algorithms that are novel *individual methods* that do not seek any additional meta-data for the network inference. Examples of such algorithms are: bLARS, ARACNE [3], CLR [2], TIGRESS [4], GENIE3 [7] etc.

Table 1 shows the performance of the bLARS algorithm along with all 29 algorithms that participated in the DREAM5 challenge. The columns Synthetic, E. coli and Yeast in the Table 1 show network-wise scores (21) of the algorithms on the respective networks. Theses algorithms are sorted in the decreasing order of their overall score, and a subset of them that are relevant to the comparison are annotated in the table. Further, the score of any *individual method* that is higher than the network wise score of the bLARS algorithm is highlighted in green. The remaining algorithms either belong to the first category, or achieve scores that are lower than the score of the bLARS algorithm. It is clear that the bLARS algorithm out-performs every other method in terms of the overall score, which measures the average inference ability of the algorithm across different datasets/organisms. This suggests that bLARS is a versatile method that not only offers a simple interpretation of the regulation, but also offers the best overall performance among currently available methods.

TABLE 1
Performance of the bLARS Algorithm Versus Performance of All the Algorithms that Participated in the DREAM5 GRN Inference Challenge

Team	Overall Score	AUPR Score	AUROC Score	Synthetic	E. coli	Yeast
bLARS	**45.596**	**66.657**	**24.535**	**123.467**	**5.841**	**7.479**
GENIE3	40.279	41.295	39.263	104.655	14.794	1.387
ANOVA	34.023	30.908	37.137	53.981	45.877	2.209
TIGRESS	31.099	41.368	20.831	87.801	4.414	1.082
REG2	28.747	44.689	12.806	86.242	0.000	0.000
META1	22.711	36.795	8.627	49.732	10.128	8.274
REG3	21.398	32.838	9.957	57.078	6.533	0.583
Team 868	20.694	31.965	9.422	56.502	3.512	2.068
Team 842	16.686	14.538	18.833	45.256	4.746	0.054
Team 861	13.397	13.781	13.013	35.514	2.075	2.601
META3	10.586	10.491	10.681	4.422	8.535	18.801
Team 799	8.887	3.076	14.699	25.945	0.709	0.009
Team 875	8.8	3.984	13.616	25.167	1.232	0.000
Team 823	8.126	5.632	10.619	22.119	2.235	0.024
META4	6.42	8.183	4.658	3.878	11.329	4.055
META5	6.332	8.337	4.327	0.076	8.687	10.233
REG6	6.026	8.924	3.128	5.451	12.131	0.497
Team 864	4.973	9.947	0.000	14.891	0.028	0.000
Team 705	2.484	2.133	2.835	2.315	1.596	3.541
Team 504	2.266	4.531	0.000	6.735	0.061	0.000
OTHER5	1.897	0.066	3.727	0.000	5.534	0.155
Team 829	1.475	2.372	0.579	0.000	3.585	0.841
Team 802	0.997	0.000	1.995	2.975	0.001	0.016
Team 756	0.331	0.173	0.489	0.043	0.000	0.949
Team 736	0.257	0.040	0.473	0.123	0.000	0.647
Team 854	0.256	0.040	0.472	0.122	0.000	0.647
Team 638	0.144	0.250	0.038	0.384	0.000	0.048
Team 784	0.035	0.003	0.068	0.000	0.000	0.105
Team 787	0	0.000	0.000	0.000	0.000	0.000
Team 821	0	0.000	0.000	0.000	0.000	0.000

It is also to be noted that the performance of the almost all the algorithms decreases from the *in silico* network to real organisms. This could be a consequence of the increasing complexity of the network from simulated network to Eukaryotic GRN. Because of this pattern, the overall score of all the algorithms is highly influenced by its performance on the *in silico* network. The *in silico* gold-standard network has far more edges (4,012) than the E. coli (2,066) or Yeast (3,940) gold-standard networks. This is in contrast with the fact that the total number of genes in the *in silico* network is only about one-third of the Yeast network. The E. coli gold-standard network is the most sparse; perhaps this is a reason for the relatively poorer performance of the bLARS algorithm.

It is observed that the bLARS algorithm does not perform as well on the E. coli data as it does on either Yeast or synthetic data. To check the possibility that E. coli network may have higher number of average regulators per gene, we ran the bLARS algorithm for 50 LARS steps (Supplementary Information Fig. 2, available online), and observed only a slight increase in the performance. We speculate that the poor performance of the bLARS algorithm on the E. Coli data is partly because of the higher degree of incompleteness of the E. coli gold-standard compared to the Yeast or the Synthetic networks.

4.4 Basis Functions in bLARS

The primary objective of the bLARS algorithm is to accommodate linear and non-linear regulations that are known to co-exist in biology. As mentioned previously, we have chosen five different functions, namely: linear, cubic, logistics up-regulation, logistics down-regulation and square-root to provide a wide range of regulation mechanisms. To demonstrate that bLARS leads to improved performance compared to using just one regulation mechanism for every gene, we ran the LARS algorithm five times, once with each of the above five functions, for 500 bootstrap runs. The results are shown in Fig. 5. It is clear from the figure that

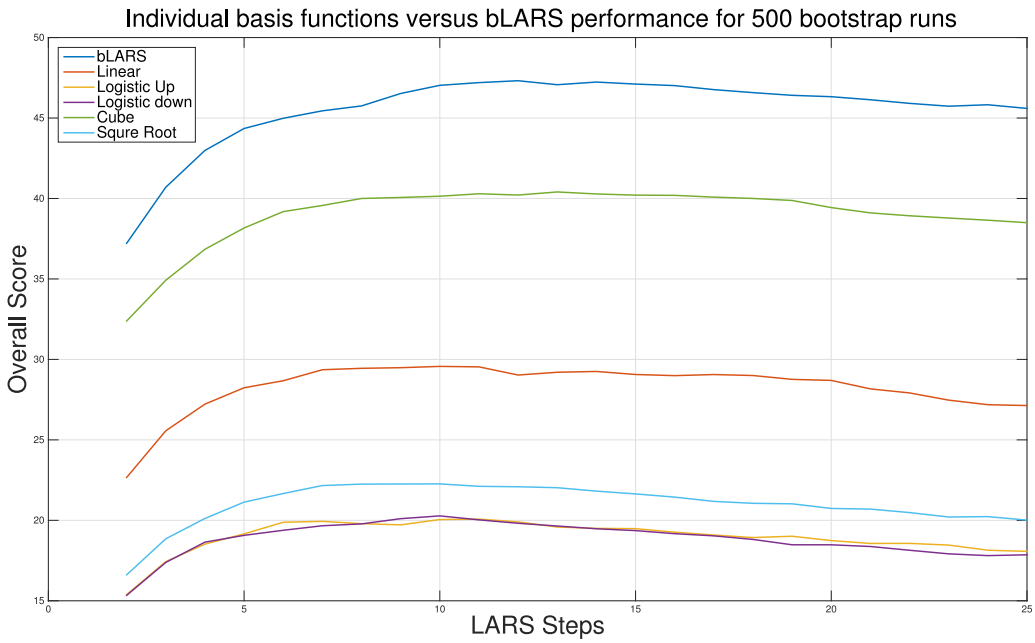

Fig. 5. bLARS performance: Individual basis functions versus bLARS for various numbers of LARS steps.

none of individual basis functions themselves has a satisfactory score compared to bLARS. The choice of the four basis functions, in addition to the linear function, is influenced by the following reasons. The logistics regulation functions were included as they occur in the kinetic modeling of transcription regulation [37]. The cubic and square-root functions were included to reflect the cases when target expression grows slower or faster than the linear regulation case. Because the rationale to use these basis functions is based on the general transcription regulation mechanism, the choice of the basis functions is agnostic to the organism and the experimental techniques used to generate the gene expression data. The choice of these functions is up to the user and the performance would only improve if more basis functions are added.

4.5 Effect of the Parameters on the Performance of bLARS

There are two parameters in bLARS, namely: the number of bootstrap runs r and the number of LARS steps L. Fig. 6 shows the effect of these two parameters on the bLARS performance. The figure shows network-wise scores to bring out the effects for each individual network inferencing problem. It is clear that the performance of bLARS is quite robust to the number of bootstrap runs so long as it is larger than certain threshold, typically around 200 runs. The dependence of the performance of bLARS on the number of LARS steps (L) is more subtle. One's intuition dictates that the performance would be optimal if the number of LARS steps is close to the true average number of regulators in the network; the performance would drop off if the number of

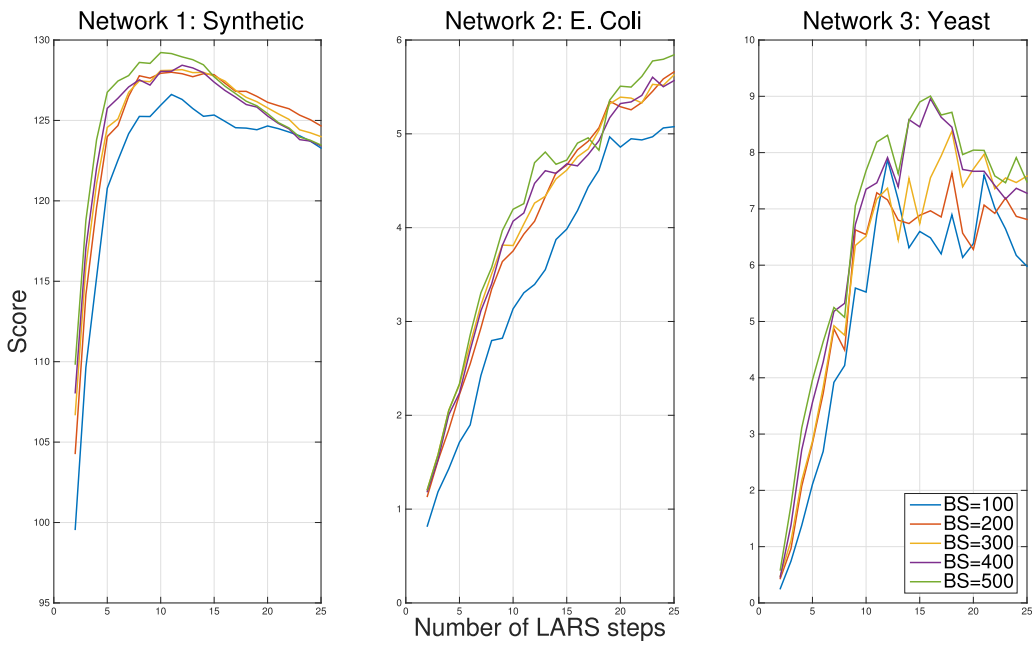

Fig. 6. Effect of the number of bootstrap runs (r) and the number of LARS steps (L) on the performance of the bLARS.

LARS steps is either too small or too large. This is apparent for the *in silico* and Yeast networks, but not for the E. Coli network. However, as shown in Fig. 5, overall performance is quite robust to the number of LARS steps as well.

5 Discussion

The network perspective for studying biological systems and disease has been gaining momentum during recent years. It is being applied to a variety of areas including the stratification of cancer tumors, and the prediction of clinical outcomes [38]. Gene regulatory networks can complement current analyses that are primarily based on protein-protein interaction (PPI) networks. Therefore, constructing GRN from experimental data is an important problem that could potentially contribute to our understanding of the basic biology of diseases.

In this work, a new regression-based algorithm named bLARS is presented to infer a GRN from steady state gene expression data. The key contribution of this method is to offer flexibility in the choice of regulation functions for each regulator-target gene pair. This distinguishes bLARS from TIGRESS which is closest to the bLARS algorithm in terms of the techniques used. Additionally, for the purpose of scoring, the bLARS algorithm uses bootstrapping while TIGRESS relies on the stability selection technique. Bootstraping and closely related subsampling techniques are commonly used in the systems biology community to address the availability of only a small number of samples. On the benchmark DREAM5 datasets, the bLARS algorithm achieves significantly higher overall scores than any other algorithm that participated in the challenge. Specifically, the bLARS algorithm performs much better than the TIGRESS algorithm demonstrating the utility of the application of nonlinear basis functions. Indeed, if one were to exclude methods that rely on perturbation data and meta-methods that combine two or more existing techniques, then among all methods only GENIE3 offers performance comparable to that of bLARS. As evident from the results, the additional perturbation information improves performance in the case of *E. Coli*, but does not offer similar performance gains in the case of *in silico* or Yeast networks. Meta-algorithms that combine multiple approaches would improve the inference even further. For example, one can combine the bLARS algorithm with other algorithms to get a new algorithm. This is why meta-algorithms are grouped into a category distinct from individual methods.

We note that the use of multiple basis functions in the bLARS algorithm results in the improved performance, but at the cost of more computation time. However, the formulation of the bLARS algorithm is very amenable to the parallel computing; thus total serial "wall clock" computing time should not be a major concern. The current Matlab implementation of the bLARS algorithm takes about 7, 15 and 13 hours on the synthetic, E. coli and Yeast networks respectively for the recommended 200 bootstrap runs on a 12 core Intel Xeon 2.67 GHz machine. If one were to run bootstraps in parallel then the computing time reduces linearly. For instance, the presented results were generated with four parallel workers; thus the total computation time was reduced to approximately one-fourth. The current implementation of the bLARS algorithm takes about twice the time of the TIGRESS algorithm, if run in the serial manner. The GENIE3 and ANOVA-η^2 algorithms are implemented in C; therefore they run faster; a C implementation of bLARS algorithm can also achieve a similar performance gain.

The problem of learning the interaction structure from data *without any prior information* is very difficult [39]. However, in many of the problems, including the problem of GRN inference, it is reasonable to assume that the interaction structure is sparse. In the context of GRN inference, the sparsity assumption means that every gene has only a small number of regulators, which seems quite reasonable. The proposed bLARS algorithm also makes the same assumption, as do several other current GRN inference algorithms. Unfortunately, the investigated problem falls under the case where the number of samples are much smaller than the feature dimension, also referred to as $n < p$ case in the statistics and machine learning literature. For such problems, theoretical guarantees of performance are available only under fairly restrictive assumption on the near-orthogonality of the columns of the data matrix X, which are hard to rationalize in the GRN inference problem. Because of this situation, the proposed bLARS algorithm is currently a heuristic and theoretical analysis is left to future work. The same is true of other GRN inference algorithms as well.

Another important issue is the validation of the inferred network. The DREAM challenge has set up a good standard framework for comparing different algorithms, but the incompleteness of the gold-standard networks is a severe limitation. The missing edges from the gold-standard networks may lead to the under-estimation of the number of true positives. As more edges are added to the gold-standard network, the predicted true positives can either increase (i.e. false positives could decrease) or stay the same. This is because the number of predicted positive edges is sum of the number of true positives and the number of false positives. Therefore, the incompleteness of the gold-standard under-estimates the performance. This is why one has to use simulated gene expression data for a known gold-standard network to compare the performance of network inference algorithms in addition to validating on biological networks. The assessment of the performance of different algorithms on real biological networks will improve in future as evidence for more gold-standard edges is gathered. In the interim, synthetic networks will complement the algorithm benchmarking experiments. Therefore, it is crucial to use GRN simulation tools that account for as many biological factors as possible. The synthetic network in the DREAM5 challenge was generated via GeneNetWeaver tool [36] that attempts to account for the network complexity and the various sources of noise present in the biological systems[1]. Further, the variability of the performance of the current state-of-the-art algorithms suggests that there is no one algorithm that performs equally well on all datasets. Therefore all of these algorithms can be thought of as providing inputs to a meta-algorithm that relies on "the wisdom of crowds" to create a consensus community network. Also, the decreasing performance of all the algorithms from synthetic network to Yeast perhaps reflects the increasing complexity of the underlying regulatory networks. Our method advances the current state of the art, but there is still a long way to go before the problem could be treated as completely solved.

6 Conclusion

In this work, we have presented a new algorithm called bLARS to infer gene regulatory networks from steady state gene expression data. Our algorithm provides for different genes to have different regulatory mechanisms. The performance of the bLARS algorithm is compared with those of several state-of-the-art GRN inference algorithms on three datasets, namely *in silico*, E. coli and Yeast gene expressions. A comparison of the results shows that bLARS has superior overall performance. The algorithm does not require additional information such as perturbation information from gene knock-down studies, and is robust to variation in its parameters. The algorithm implementation and bench mark DREAM data is also available as supplementary information, available online.

Acknowledgments

The authors would like to thank Burook Misganaw for his feedback and discussions. This research was supported by the US National Science Foundation under ECCS Awards #1001643 and #1306630; the Cancer Prevention and Research Institute of Texas (CPRIT) under grant RP140517; the Jonsson Family Graduate Fellowship; and the Cecil H. and Ida Green Endowment at the University of Texas at Dallas.

References

[1] D. Marbach, C. Costello James, R. Küffner, et al. "Wisdom of crowds for robust gene network inference," *Nature Methods*, vol. 9, no. 8, pp. 796–804, 2012.

[2] J. Faith Jeremiah, B. Hayete, T. Thaden Joshua, et al. "Large-scale mapping and validation of Escherichia coli transcriptional regulation from a compendium of expression profiles," *PLoS Biol.*, vol. 5, no. 1, p. e8, Jan. 2007.

[3] A. Margolin, I. Nemenman, K. Basso, et al. "ARACNE: An algorithm for the reconstruction of gene regulatory networks in a mammalian cellular context," *BMC Bioinformatics*, vol. 7, p. S7, Jan. 2006.

[4] A.-C. Haury, F. Mordelet, P. Vera-Licona, and J.-P. Vert, "TIGRESS: Trustful inference of gene REgulation using stability selection," *BMC Syst. Biol.*, vol. 6, no. 1, p. 145, 2012.

[5] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Using Bayesian networks to analyze expression data," *J. Comput. Biol.*, vol. 7, no. 3/4, pp. 601–20, Jan. 2000.

[6] J. Butte Atul and I. S. Kohane, "Unsupervised knowledge discovery in medical databases using relevance networks," in *Proc. AMIA Symp.*, Jan. 1999, pp. 711–715.

[7] Vân Anh 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, Jan. 2010.

[8] R. Küffner, T. Petri, P. Tavakkolkhah, L. Windhager, and R. Zimmer, "Inferring gene regulatory networks by ANOVA," *Bioinformatics*, vol. 28, no. 10, pp. 1376–82, May 2012.

[9] M. Rowicka, A. Kudlicki, P. T. Benjamin, and Z. Otwinowski, "High-resolution timing of cell cycle-regulated gene expression," *Proc. Nat. Acad. Sci. United States of America*, vol. 104, no. 43, pp. 16892–7, Oct. 2007.

[10] DREAM5. (2010). DREAM5 challenge website. Data available to download [Online]. Available: http://wiki.c2b2.columbia.edu/dream/index.php?title=D5c4

[11] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression," *Ann. Statist.*, vol. 32, no. 2, pp. 407–499, 2004.

[12] B. E. Michael, T. S. Paul, O. B. Patrcik, and D. Botestein, "Cluster analysis and display of Genome-wide expression patterns," *Proc. Nat. Acad. Sci. United States of America*, vol. 95, no. 22, pp. 14863–14868, 1998.

[13] B. Zhang and S. Horvath, "A general framework for weighted gene Co-expression network analysis," *Statistical Appl. Genetics Molecular Biol.*, vol. 4, no. 1, pp. 1–45, 2005.

[14] P. Langfelder and S. Horvath, "WGCNA: An R package for weighted correlation network analysis," *BMC Bioinformat.*, vol. 9, no. 1, p. 559, Jan. 2008.

[15] A. J. Butte and I. S. Kohane, "Mutual information relevance networks: Functional genomic clustering using pairwise entropy measurements," in *Proc. Pacific Symp. Biocomput.*, Jan. 2000, pp. 418–429.

[16] J. Yu, V. Anne Smith, P. W. Paul, J. H. Alexander, and D. J. Erich, "Advances to Bayesian network inference for generating causal networks from observational biological data," *Bioinformatics*, vol. 20, no. 18, pp. 3594–603, Dec. 2004.

[17] M. C. Thomas and A. T. Joy, *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing*. New York, NY, USA: Wiley 2006.

[18] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Roy. Statistical Soc.: Series B (Statistical Methodol.)*, vol. 59, no. 1, pp. 267–288, 1996.

[19] E P V. Someren, B L T. Vaes, W T. Steegenga, et al. "Least absolute regression network analysis of the murine osteoblast differentiation network," *Bioinformatics*, vol. 22, no.4, pp. 477–84, Feb. 2006.

[20] J. Slawek and T. Arodź, "ADANET: Inferring gene regulatory networks using ensemble classifiers," in *Proc. ACM Conf. Bioinformat., Comput. Biol. Biomed.*, New York, NY, USA, 2012, pp. 434–441.

[21] X. Zhang, K. Liu, Z.-P. Liu, et al. "NARROMI: A noise and redundancy reduction technique improves accuracy of gene regulatory network inference," *Bioinformatics*, vol. 29, no. 1, pp. 106–13, Jan. 2013.

[22] M. Bansal, V. Belcastro, A. Ambesi-Impiombato, and D. di Bernardo, "How to infer gene networks from expression profiles," *Molecular Syst. Biol.*, vol. 3, no. 78, p. 78, Jan. 2007.

[23] G. Karlebach and R. Shamir, "Modelling and analysis of gene regulatory networks," *Nature Rev. Molecular Cell Biol.*, vol. 9, no. 10, pp. 770–80, Oct. 2008.

[24] C. Y. William, E. R. Adrian, and Y. Y. Ka, "Fast Bayesian inference for gene regulatory networks using ScanBMA," *BMC Syst. Biol.*, vol. 8, no. 1, p. 47, Apr. 2014.

[25] M. Bansal, G. Della Gatta, and D. di Bernardo, "Inference of gene regulatory networks and compound mode of action from time course gene expression profiles," *Bioinformatics*, vol. 22, no. 7, pp. 815–822, Apr. 2006.

[26] V. A. Huynh-Thu and G. Sanguinetti, "Combining tree-based and dynamical systems for the inference of gene regulatory networks," *Bioinformatics*, vol. 31, no. 10, pp. 1614–1622, 2015.

[27] S. G. Timothy and J. F. Jeremiah, "Reverse-engineering transcription control networks," *Phys. Life Rev.*, vol. 2, no. 1, pp. 65–88, Mar. 2005.

[28] E. Segal and J. Widom, "From DNA sequence to transcriptional behaviour: A quantitative approach," *Nature Rev. Genetics*, vol. 10, no. 7, pp. 443–56, Jul. 2009.

[29] M. Yuan and Yi Lin, "Model selection and estimation in regression with grouped variables," *J. Roy. Statistical Soc.: Series B (Statistical Methodol.)*, vol. 68, pp. 49–67, 2006.

[30] N. Simon, J. Friedman, T. Hastie, and R O B Tibshirani, "A Sparse-group lasso," *arXiv*, pp. 1–19, 2007.

[31] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*, Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Boca Raton, FL, USA: CRC Press, 1994.

[32] S. Wang, B. Nan, S. Rosset, and Ji Zhu. "Random Lasso," *Ann. Appl. Statist.*, vol. 5, no. 1, pp. 468–485, Mar. 2011.

[33] N. Meinshausen and P. Bühlmann, "Stability selection," *J. Roy. Statistical Soc.: Series B*, vol. 72, no. 4, pp. 417–473, 2009.

[34] DREAM. (2006). DREAM dialogue for reverse engineering assessments and methods [Online]. Available: http://www.the-dream-project.org/

[35] Z. Zhang, J. Wang, and H. Zha, "Adaptive manifold learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 2, pp. 253–65, Feb. 2012.

[36] T. Schaffter, D. Marbach, and D. Floreano, "GeneNetWeaver: In silico benchmark generation and performance profiling of network inference methods," *Bioinformatics*, vol. 27, no. 16, pp. 2263–2270, 2011.

[37] U Alon, *An Introduction to Systems Biology: Design Principles of Biological Circuits*, Chapman & Hall/CRC Mathematical and Computational Biology. Boca Raton, FL, USA: CRC Press, 2006.
[38] M. Hofree, P. S. John, H. Carter, A. Gross, and T. Ideker, "Network-based stratification of tumor mutations," *Nature Methods*, vol. 10, no. 11, pp. 1108–15, Nov. 2013.
[39] J. Gonccalves and S. Warnick, "Necessary and sufficient conditions for dynamical structure structure reconstruction of LTI networks," *IEEE Trans. Automat. Control*, vol. 53, no. 7, pp. 1670–1674, Aug. 2008.

Nitin Singh received the MS degree in bioengineering from the University of Texas at Dallas in 2012. He is currently working toward the PhD degree in bioengineering at the University of Texas at Dallas, Richardson, TX. His research interest lies in the machine learning and its application to systems biology.

Mathukumalli Vidyasagar received the BS, MS, and PhD degrees in electrical engineering from the University of Wisconsin, Madison, WI, in 1965, 1967, and 1969, respectively. For the next 20 years, he taught at Marquette University (1969-1970), Concordia University (1970-1980) and the University of Waterloo (1980-1989). In 1989, he returned to India as the director in the newly created Center for Artificial Intelligence and Robotics (CAIR), Bangalore, under the Ministry of Defense, Government of India. In 2000, he moved to the Indian private sector as an executive vice president in India's largest software company, Tata Consultancy Services (TCS), where he created the Advanced Technology Center, an industrial R&D laboratory of around 80 engineers. In 2009, he retired from TCS and joined the Erik Jonsson School of Engineering and Computer Science, University of Texas at Dallas, Richardson, TX, as a Cecil and Ida Green chair in systems biology science. In March 2010, he was named as the Founding head of the newly created Bioengineering Department and served in that capacity until June 2013. His current research interests are in stochastic processes and stochastic modeling, and their application to problems in computational biology. He received a number of awards in recognition of his research, including the IEEE Control Systems ("Field") Award in 2008, the Fellowship of the Royal Society, United Kingdom, in 2012, the ASME Rufus Oldenburger Medal in 2012, and the AACC John R. Ragazzini Education Award in 2013. He is a life fellow of the IEEE.