Received: 15 June 2021 | Revised: 29 October 2021 | Accepted: 10 January 2022

DOI: 10.1002/rnc.6044

SPECIAL ISSUE ARTICLE

WILEY

Scalable reverse-engineering of gene regulatory networks from time-course measurements

Francesco Montefusco[1] | Anna Procopio[2] | Declan G. Bates[3] | Francesco Amato[4] | Carlo Cosentino[2]

[1] Department of Biomedical Sciences, Universita degli Studi di Sassari, Sassari, Italy

[2] School of Computer and Biomedical Engineering, Universita degli Studi Magna Græcia di Catanzaro, Catanzaro, Italy

[3] School of Engineering, University of Warwick, Coventry, UK

[4] Department of Electrical Engineering and Information Technology, Universita degli Studi di Napoli Federico II, Napoli, Italy

Correspondence

Carlo Cosentino, School of Computer and Biomedical Engineering, Universita degli Studi Magna Græcia di Catanzaro, Viale Europa, Campus di Germaneto, 88100 Catanzaro, Italy.
Email: carlo.cosentino@unicz.it

Funding information

Regione Calabria, Grant/Award Number: Regional Project POR Calabria FESR/FSE 2014-2020; Università degli Studi di Sassari, Grant/Award Number: Fondo di Ateneo per la ricerca 2020

Abstract

Topological inference of biological interaction networks from experimental data is a fundamental research topic in the broad area of Systems Biology. Several algorithms presented in the literature have been devised in order to infer the topology of a network from time-course data. The present work introduces a novel method for reverse-engineering gene regulatory networks from time-course experiments, which combines the instrumental variables technique for the identification of dynamical systems with a regularization strategy for dealing with over-parametrized systems. Differently from least squares methods, the proposed approach can explicitly address the bias and nonconsistency issues that arise when dealing with time-course measurements, thus yielding improved performance with respect to methods designed for steady-state data. Moreover, the devised approach, which has been named RIVA (Reverse-engineering of biological networks via Instrumental VAriables), can simultaneously exploit multiple time-series, thus enabling one to get improved results by collecting and exploiting data from multiple experiments, and is computationally efficient, thus it can be also applied to large-scale (in the order of thousands of nodes) networks. To analyze the applicability and effectiveness of RIVA, we performed several tests, both with simulated data and with experimental data, and compared the results against other state-of-the-art inference methods designed for time-series data.

KEYWORDS

biological systems, network inference, system identification

1 | INTRODUCTION

A key challenge in the field of Systems Biology is the reverse-engineering (or inference) of the topology of biological interaction networks at the molecular level using the measurements of the nodes' expression under different experimental conditions, for example, external perturbations (drugs, signaling molecules, pathogens), or changes in environmental conditions (change in the concentration of nutrients, or in the temperature level). Most biological processes are controlled by gene regulatory networks, thus it is paramount to understand how the expression of each gene is influenced by the other ones over time. To this aim, several algorithms have been proposed to infer a gene regulatory network from

Francesco Montefusco and Anna Procopio contributed equally to this study.

time-course experimental data using different methodologies, for example, TSNI,[1] based on regression, or the Inferelator,[2] based on meta predictors. There are also algorithms based on statistical methods, such as Bayesian networks[3] and Mutual Information theory,[4] which are currently the most widely used tools for network inference in biology. Such approaches usually require large datasets and/or assume that the measurement samples are independent. In many situations, however, only a small number of experimental data points may be available, and the assumption of independent samples is clearly not satisfied for measurements of the same gene expression at two consecutive time-points. On the other hand, methods based on the identification of ordinary differential equation (ODE) models of biological networks, have been shown to provide an effective alternative to statistical approaches.[5-11] A common limitation to the practical application of the latter class of methods is the detrimental effect of measurement noise. Indeed, the performance of the dynamical model-based approaches have been shown to degrade significantly in the presence of even limited amounts of noise in the measurement data,.[6,8,12,13] In Reference 10, the authors propose an approach for network inference that takes into account noise and nonlinear dynamics; this method, however, seems to be only suitable for small-scale networks. An attempt to address the problem of noise, using the Constrained Total Least Squares (CTLS) algorithm, showed promising results,[14] but involves a level of computational complexity that limits its application to relatively small-scale networks as well.

In this work, we describe an alternative approach which explicitly takes into account the effect of measurement noise within the inference process while minimizing computational overheads, so that large-scale networks can be inferred. In fact, the RIVA algorithm introduced in this work can be effectively applied to networks of thousand nodes, thus it is possible to overcome the computational limitations of the aforementioned methods. Our approach exploits the Instrumental Variables (IV) identification technique p. 486 of Reference 15, to avoid the introduction of bias and nonconsistency due to measurement noise, which affects the estimation of the parameters of a dynamical system when using the standard Least Squares (LS) method.

Network inference methods typically tackle the problem in two steps: first, the full model structure is identified, thus assigning a weight to each putative edge between every pair of node; subsequently, the edges are ranked according to some criterion based on the computed edge weights. This strategy also poses the challenging problem of selecting the true edges among all the possible candidates.

This is a feature selection problem, which can be tackled by means of some regularization strategy, such as the regularized LS (RLS) technique, called ridge regression in statistics p. 49 of Reference 15, to deal with over-parametrized models and with the related problem of under-determination of the model structure. These techniques introduce a penalty on the norm of the optimization variables, thus reducing the variance of the estimated parameters, which are subject to a shrinking around the true value. Consequently it is easier to identify the parameters with true values equal to zero and remove the corresponding terms from the estimated model.

The proposed approach has been first evaluated against a simulated dataset, generated via GeneNetWeaver (GNW), an open-source tool for the automatic generation of in silico gene networks and reverse engineering benchmarks.[16-18] Although our approach is based on the use of linear models of biological interaction networks, the in silico networks used for its evaluation are nonlinear. The proposed algorithm is also tested on experimental micro-array data measuring the time-course expression of a regulatory subnetwork of the cell cycle in *S. cerevisiae*. Finally, the results obtained by RIVA are compared with those yielded by other two state-of-the-art network inference algorithms, namely dynGenie3[19] and BINGO.[20] The results show that the proposed approach yields in most cases superior performance, while requiring a much lower computational burden and, thus, being more scalable to large networks.

2 | METHODS

The RIVA inference algorithm is based on the combination of the IV and RLS methods. In this section we introduce the machinery of the proposed technique. Subsequently, the generation of the different datasets used to evaluate the performance of RIVA is illustrated. Moreover, we briefly recall the main features of dynGENIE3 and BINGO.

Our approach is based on the use of Linear Time-Invariant (LTI) models of the network dynamics, in the form

$$\dot{x}(t) = Ax(t) + Bu(t), \tag{1}$$

where $x(t) = (x_1(t), \dots, x_n(t))^T \in \mathbb{R}^n$, the state variables $x_i,\ i = 1, \dots, n$, denote the quantities of the different compounds present in the system (e.g., mRNA concentrations for gene expression levels), $A \in \mathbb{R}^{n\times n}$ is the state transition matrix and $B \in \mathbb{R}^{n\times m}$ is a matrix that determines the direct targets of external perturbations $u(t) = (u_1(t), \dots, u_m(t))^T \in \mathbb{R}^m$ (e.g., drugs, overexpression, or downregulation of specific genes), which are typically induced during in vitro experiments.

Although the dynamics of biological networks are typically nonlinear, the use of linear models in network inference algorithms is fairly common and is justified by the fact that our goal is not to identify an accurate predictive model of the state trajectory of the biological system, but rather to recover the functional dependencies between the state variables. Furthermore, the use of a linear model relies on the assumption that the system is at steady state and the it undergoes a relatively small perturbation. Note that the derivative (and therefore the evolution) of x_i at time t is directly influenced by the value $x_j(t)$ if $\{A\}_{ij} \neq 0$. Moreover, the type of influence (i.e., whether the source node promotes or inhibits the expression of the target node) and the strength of this interaction can be associated with the sign and magnitude of the element $\{A\}_{ij}$, respectively. Thus, if we consider the state variables as quantities associated with the nodes of a network, matrix A can be considered as a compact numerical representation of the network topology, that is, a weighted adjacency matrix. Therefore, the network topology can be inferred by identifying the dynamical system (1) from the experimental measurements.[5,7,21,22] In particular, while the adjacency matrix A provides information about the edges of the network, the input matrix B can be associated to the experimental perturbations. Each column of B is associated to a different type of perturbation that can be imposed to the network: each column has one or more nonzero entries defining which nodes are direct targets of that specific perturbation. If the input vector for the pth experiment is defined as

$$u^{(p)}(\cdot) = \begin{pmatrix} 0 & \dots & 0 & u_p(\cdot) & 0 & \dots & 0 \end{pmatrix}^T,$$

the effect of the pth perturbation is determined by the pth column of B and by $u_p(\cdot)$. Note that, using the same model, it is also straightforward to deal with experiments where a combination of multiple perturbations is applied, by allowing more than one element of $u^{(p)}$ to be nonzero.

Given the perturbation inputs and the corresponding measurements of the full state of the network sampled at $h+1$ time points, that are $\bar{u}^{(p)}(t_k)$ and $\overline{x}^{(p)}(t_k)$, $k = 0, \dots, h$, for each perturbation experiment, $p = 1, \dots, m$, the problem can be reformulated in the discrete-time domain as

$$\overline{x}^{(p)}(k+1) = A_d \overline{x}^{(p)}(k) + B_d\, \bar{u}^{(p)}(k), \tag{2}$$

where, for the sake of brevity, we adopt the notation $\overline{x}^{(p)}(k) := \overline{x}^{(p)}(t_k)$. Note that we assume equispaced samples in time; in the case of non-equispace sampling, the data have to be interpolated (e.g., through a cubic spline) to generate the missing samples.

The matrices A_d and B_d are the discretized counterparts of the time-continuous A and B in (1). Using a discretization method, for example, the zero-order-hold, it is easy to demonstrate that $A_d \approx I + A\,T_s$ and $B_d \approx B\,T_s$, where the approximation is closer when the sampling time T_s is sufficiently small with respect to the dominant time-constant of the time-continuous system (see the appendix in Reference 8). Thus, the topology of the network and the targets of perturbations can, at least in principle, be inferred also by A_d and B_d. For the sake of simplicity, in the following we will use the symbols A and B to denote A_d and B_d. Let

$$\overline{X}_p^+ = \begin{pmatrix} \overline{x}^{(p)}(1) & \dots & \overline{x}^{(p)}(h) \end{pmatrix} \in \mathbb{R}^{n\times h},$$
$$\overline{X}_p = \begin{pmatrix} \overline{x}^{(p)}(0) & \dots & \overline{x}^{(p)}(h-1) \end{pmatrix} \in \mathbb{R}^{n\times h},$$
$$\bar{U}_p = \begin{pmatrix} \bar{u}^{(p)}(0) & \dots & \bar{u}_n^{(p)}(h) \\ 0 & \dots & 0 \end{pmatrix} \in \mathbb{R}^{n\times h},$$

The identification model reads

$$\Xi = \hat{\Theta}\Omega, \tag{3}$$

where

$$\Xi := \begin{pmatrix} \overline{X}_1^+ & \dots & \overline{X}_m^+ \end{pmatrix}, \quad \Theta := \begin{bmatrix} \hat{A} & \hat{B} \end{bmatrix}, \quad \Omega := \begin{pmatrix} \overline{X}_1 & \dots & \overline{X}_m \\ \bar{U}_1 & \dots & \bar{U}_m \end{pmatrix},$$

$\hat{A}$ and $\hat{B}$ are the estimates of the matrices of system (2). Regarding the perturbation inputs, note that it is not required to know the absolute value of the signal $u^{(p)}(\cdot)$, but only a normalized profile, since the scaling factor is embedded in the estimate of $\hat{B}$. For instance, if the pth perturbation experiment consists of exposing the cell to a constant amount of metabolite/drug during the first l time steps, this can be easily obtained by setting

$$u^{(p)}(k) = \begin{cases} 1 & \text{for } 0 < k < l \\ 0 & \text{for } k > l \end{cases}. \tag{4}$$

In general, the direct targets of the perturbation are unknown; however, when using perturbations that directly influence the expression of one or more given nodes (e.g., gene knock-down or knock-out), this information can be readily exploited in the present modeling framework, by a priori assigning the zero pattern of the corresponding column in the $\hat{B}$ matrix.

The computational burden can be reduced by decomposing row-wise the identification problem, that is the ith row Θ_{i*} can be separately identified using the multiple regression model

$$Y = Z\beta + \varepsilon, \tag{5}$$

where $Y := \Xi_{i*}^T \in \mathbb{R}^{hm}$, $Z = \Omega^T \in \mathbb{R}^{hm\times(n+m)}$, $\beta = (\Theta_{i*})^T \in \mathbb{R}^{n+m}$ and $\varepsilon \in \mathbb{R}^{hmh}$ is the measurement noise, which admits the standard LS solution

$$\hat{\beta}_{LS} = \left(Z^T Z\right)^{-1} Z^T Y. \tag{6}$$

The matrix $Z^\dagger := \left(Z^T Z\right)^{-1} Z^T$ is the (Moore–Penrose) pseudo-inverse of Z. Note that, to compute $Z^\dagger$, it is necessary that $Z^T Z$ is invertible; this is always possible if the $n + m$ columns of Z (the regression vectors) are linearly independent, which requires $(h\,m) \geq n + m$, that is, one should have at least as many measurements as regression coefficients. Note that, in theory, satisfaction of the latter inequality does not guarantee the invertibility of $Z^T Z$; however, this is always true in practice, because the presence of noise renders equal to zero the probability of exact singularity. On the other hand, a nonsingular $Z^T Z$ does not guarantee an accurate solution: when $Z^T Z$ is nearly singular the effects of noise and round-off errors on the estimated coefficients β are very high, undermining the chances to recover the true values.

Note that, when the regression is performed on time-series measurements, the regressor matrix is not made up of independent columns: if we consider a single experiment, the columns of Z are sampled points of a single-state trajectory, thus the correlation of any two consecutive measurements takes a (nonzero) value, which depends on the type of perturbation, the system dynamics and the sampling time. If the dynamics of the system are smooth and slow, then $x^{(p)}(k)$ can be approximated by a linear combination of its values at the previous steps, $x^{(p)}(k-1), \ldots, x^{(p)}(0)$. This implies that the columns of $Z^T Z$ are almost linearly dependent, which renders the LS solution highly sensitive to noise and round-off errors. A further difference, with respect to the standard LS regression problem, is that the regressor variables are not deterministic, since they are affected by noise: Ω^T contains measured variables $\overline{x}^{(p)}(k)$, including noise. For the above reasons, the parameter estimates are generally *biased* and *nonconsistent*. Biased parameter estimates provide an average value that systematically deviates from the optimal value; nonconsistency, on the other hand, means that the bias does not approach zero even when the number of samples $h \to \infty$.

It is worth noting that two common strategies for improving the identification results when using time-series measurements, that is, increasing the number of measurements by either reducing the sample time or by considering a longer time interval, might be useless in this context: indeed, having $x^{(p)}(k)$ too close in time to $x^{(p)}(k-1)$ increase the correlation between the regression vectors. On the other hand, taking additional measurements after the state has reached the steady state, also introduces new regression vectors that are linearly dependent on the previous ones (by definition, at steady-state $x^{(p)}(k) = x^{(p)}(k-1)$). Hence, the only chance to improve the inference performance is by exploiting data from many different experiments, possibly using different perturbation inputs, which affect different nodes of the network, in order to excite all the dynamics of the network.

2.1 Regression with IV

The IV technique (see Reference 15, p. 486) will be exploited to deal with the issue of nonconsistent parameter estimates discussed above. The main machinery of this method is based on the use of a matrix $V \in \mathbb{R}^{mh\times(n+m)}$ of the same dimensions of the regressors matrix Z. The columns of V are named *instrumental variables* and are selected in such a way that they are uncorrelated with the noise, that is $V^T\varepsilon = 0$. Multiplying (5) by V^T yields

$$V^TY - V^TZ\beta = V^T\varepsilon = 0,$$

thus

$$V^TY = V^TZ\beta. \tag{7}$$

Equation (7) admits the solution

$$\hat{\beta}_{\mathrm{IV}} = \left(V^TZ\right)^{-1}V^TY. \tag{8}$$

The IV estimate (8) is equivalent to the estimate of the standard least-square solution (6) if $V = Z$. However, the columns of Z cannot be used as IV, since Z is correlated with the noise ($Z^T\varepsilon \neq 0$), therefore the key point, when using an approach based on IVs, is how to choose a suitable matrix V. In the following, we illustrate the procedure adopted in this work to compute the matrix V.

Ideally, the IVs should be highly correlated with the regressors in order to make the variance error small.[15] Effective *IVs* would be the true (i.e., noncorrupted by noise) values of $x^{(p)}(k)$, however, they are unknown. A suitable strategy is to compute an approximation of such values by filtering the measured $x^{(p)}(k)$ through the process model, which is the idea underpinning the algorithm detailed below.

1. Compute the standard LS estimate, $\overline{\Theta}_{\mathrm{LS}}$, solving (6) for each row of (3).
2. Check the stability of the identified model (3) with the parameter set $\overline{\Theta}_{\mathrm{LS}}$ and compute the simulated data $\hat{x}^{(p)}(k)$: if $\bar{A}_{\mathrm{LS}}$ is stable, then simulate the time evolution of the identified model $\overline{\Theta}_{\mathrm{LS}}$ for generating $\hat{x}^{(p)}(k)$, otherwise exploit $\overline{\Theta}_{\mathrm{LS}}$ for getting the one-step-ahead prediction of $\hat{x}^{(p)}(k)$ from $\hat{x}^{(p)}(k-1)$, for each k.
3. Construct V using the simulated data $\hat{x}^{(p)}(k)$

   $$V := \begin{pmatrix} \hat{X}^{(1)} & \dots & \hat{X}^{(m)} \\ \bar{U}_1 & \dots & \bar{U}_m \end{pmatrix}^T,$$

   where

   $$\hat{X}^{(p)} = \begin{pmatrix} \hat{x}^{(p)}(0) & \dots & \hat{x}^{(p)}(h-1) \end{pmatrix} \in \mathbb{R}^{n\times h}, \quad p = 1, \dots, m,$$

   and compute an IV estimate of the model parameters, $\overline{\Theta}_{\mathrm{IV}}$, by solving (8).
4. Compute new simulated data $\hat{x}^{(p)}_{\mathrm{IV}}(k)$ as in step 2, using model (2) with the parameter set $\overline{\Theta}_{\mathrm{IV}}$, and replace $\hat{x}^{(p)}(k)$ in P3 with $\hat{x}^{(p)}_{\mathrm{IV}}(k)$.
5. Compute the error matrix

   $$E = \Xi - \overline{\Theta}_{\mathrm{IV}}\Omega \in \mathbb{R}^{n\times mh}.$$

   Define the vectors of residuals as

   $$e^{(p)}_{\mathrm{IV}} = (e^{(p)}(1)^T, \dots, e^{(p)}(h)^T)^T \in \mathbb{R}^{nh}, \quad p = 1, \dots, m,$$

   where $e^{(p)}(k) = E_{*i} \in \mathbb{R}^n$ is the ith column of E, where $i = p \cdot k$ for $k = 1, \dots, h$.
6. Construct an autoregressive (AR) model for the residuals to extract the remaining information from e_{IV}. For each $p = 1, \dots, m$ we have the following model,

   $$e^{(p)}(k) + a^{(p)}_1\, e^{(p)}(k-1) + \dots + a^{(p)}_l\, e^{(p)}(k-l) = v^{(p)}(k), \tag{9}$$

where $v^{(p)}(k) \in \mathbb{R}^n$ is white noise and l is the dynamic order of the AR model. If we introduce the *backward shift operator* q^{-1}

$$q^{-1}e^{(p)}(k) = e^{(p)}(k-1),$$

and

$$L^{(p)}(q) = 1 + a_1^{(p)}\, q^{-1} + \cdots + a_l^{(p)}\, q^{-l},$$

then Equation (9) can be rewritten as

$$e^{(p)}(k) = \frac{1}{L^{(p)}(q)}\, v^{(p)}(k).$$

Exploiting the regression model

$$X_e^{(p)} = Z_e^{(p)}\, a^{(p)} + v^{(p)},$$

where

$$\begin{aligned} X_e^{(p)} &= (e^{(p)}(h), \ \dots\ , e^{(p)}(2))^T \in \mathbb{R}^{n(h-1)}\ , \\ v^{(p)} &= (v^{(p)}(h), \ \dots\ , v^{(p)}(2))^T \in \mathbb{R}^{n(h-1)}\ , \\ a^{(p)} &= (-a_1^{(p)}, \ \dots\ , -a_l^{(p)})^T \in \mathbb{R}^{l}, \end{aligned}$$

and

$$Z_e^{(p)} = \begin{pmatrix} e^{(p)}(h-1) & \dots & e^{(p)}(h-l) \\ \vdots & \ddots & \vdots \\ e^{(p)}(h-l-1) & \dots & e^{(p)}(1) \\ \vdots & \ddots & 0 \\ e^{(p)}(1) & 0 & 0 \end{pmatrix} \in \mathbb{R}^{(h-1)\times l},$$

compute the LS solution to find the optimal coefficients $a^{(p)}$ of the filters $L^{(p)}$, $p = 1, \ \dots\ , m$,

$$\hat{a}_{\mathrm{LS}}^{(p)} = \left(Z_e^{(p)T} Z_e^{(p)}\right)^{-1} Z_e^{(p)T} X_e.$$

7. Filter the $\hat{x}_{\mathrm{IV}}^{(p)}(k)$ computed at step 4. with the filter $\hat{L}^{(p)}(q)$ estimated at step 6,

$$\hat{x}_{IV-L}^{(p)}(k) = \hat{L}^{(p)}(q)\hat{x}_{\mathrm{IV}}^{(p)}(k). \tag{10}$$

8. Construct the new IV matrix V_L from the filtered measurements $\hat{x}_{IV-L}^{(p)}(k)$ obtained at step 7,

$$V_L := \begin{pmatrix} \hat{X}_L^{(1)} & \dots & \hat{X}_L^{(m)} \\ \bar{U}_1 & \dots & \bar{U}_m \end{pmatrix}^T,$$

where

$$\hat{X}_L^{(p)} = \left(\hat{x}_{IV-L}^{(p)}(0) \quad \dots \quad \hat{x}_{IV-L}^{(p)}(h-1)\right) \in \mathbb{R}^{n\times h},$$

for $p = 1, \ \dots\ , m$.

9. Compute the new model parameters $\overline{\Theta}_{\text{IV}-L}$ by solving (8) with the IV matrix V_L. Update $\hat{x}^{(p)}(k)$ in step 3 with the new data obtained by the simulating model (2) with the parameter set $\Theta_{\text{IV}-L}$. Repeat steps 3–9 until the values of the IV solution converges (convergence is usually very fast—in general three iterations are sufficient).

2.2 | Regularization of IV regression

Recall that the Hessian matrix $H = Z^T Z$ in the LS problem solution (6) has to be well conditioned in order to obtain accurate parameter estimates. It is well known that the probability of ill-conditioning increases with the matrix dimension and that it may lead to large variances of the parameter estimates. The condition number of the Hessian matrix, χ, can be defined by the eigenvalue spread of the matrix by the formula $\chi = \frac{\lambda_{\max}}{\lambda_{\min}}$, where $\lambda_{\max}$ and $\lambda_{\min}$ are the largest and smallest eigenvalue of H, respectively. A method for controlling the condition number of the Hessian is called *ridge regression* (see Reference 15, pp. 49–53). It consists of adding a small term α to all diagonal entries of the Hessian, $(Z^T Z + \alpha I)$. Consequently, the eigenvalues of H are modified, in particular the large eigenvalues ($\lambda_i \gg \alpha$) are not significantly influenced, whereas the small ones ($\lambda_i \ll \alpha$) become approximately equal to α. Therefore the condition number of the Hessian becomes $\chi_{\text{reg}} = \frac{\lambda_{\max}}{\alpha}$, where χ_{reg} is the regularized eigenvalue spread. Thus, the solution of (5), using the RLS approach, yields the parameter estimate

$$\hat{\beta}_{RLS} = \left(Z^T Z + \alpha I\right)^{-1} Z^T \Xi_i, \tag{11}$$

where $I \in \mathbb{R}^{mh \times mh}$ is the identity matrix. In the present work, the method of generalized cross-validation (GCV) has been applied to calculate a good estimate of α on the basis of the data (see Reference 23). In particular, α is set to the value that minimizes the function

$$F(\alpha) = \frac{\frac{1}{mh} \| (I - G(\alpha)) Y \|^2}{\left(\frac{1}{mh} \text{Trace}\,(I - G(\alpha))\right)^2}, \tag{12}$$

where

$$G(\alpha) = Z\left(Z^T Z + mh\alpha I\right)^{-1} Z^T. \tag{13}$$

The RIVA algorithm is a regularized version of the regression with IV approach presented in the previous section: the estimation of each row of Θ is computed by applying the formula

$$\hat{\beta}_{\text{RIVA}} = \left(V^T Z + \alpha I\right)^{-1} V^T Y. \tag{14}$$

Also in this case, the optimal α is computed via (12) and (13), where Z^T is replaced by V^T. The computation of V is performed using the iterative procedure illustrated in Section 2.1. Also the estimation of the coefficients of the AR filter in (9) is performed through RLS.

2.3 | Edges selection

Independently of the method selected for the estimation of the connectivity matrix, all of the entries of the estimated $\hat{A}$ are in general nonzero, whereas biological networks typically exhibit sparse connectivity, that is, the average number of connections per node is much lower than the total number of nodes. Before using the estimated $\hat{A}$ for network inference, we perform a normalization of the matrix elements; the normalization is obtained by dividing each entry by the norm of the corresponding row and column. This renders the algorithm more robust against significant magnitude differences in the expression of the nodes of the network. The normalized estimated adjacency matrix $\tilde{A}$ is

$$\tilde{A}_{ij} = \frac{\hat{A}_{ij}}{\left(\|\hat{A}_{*j}\| \cdot \|\hat{A}_{i*}\|\right)^{1/2}}. \tag{15}$$

Finally, this normalized estimated matrix is translated into an inferred network, by sorting the entries in descending order by the absolute magnitude: correspondingly, we obtain a sorted list of edges associated to the weights of the normalized connectivity matrix. The elements at the top of the list will correspond to high-confidence predictions, that is, edges with high probability of being true positives. The edges related to self-interactions between the nodes of the network were not considered in the ranking list.

2.4 | Generation of the artificial datasets

The artificial dataset for the assessment of the proposed algorithm has been generated by means of GeneNetWeaver (GNW),[24] an open-source tool for in silico benchmark generation and performance profiling of network inference methods.[16] The gene network is modeled by the following ODE system:

$$\dot{x}_i(t) = m_i f_i(y) - \lambda_i^{\mathrm{RNA}} x_i, \tag{16}$$

$$\dot{y}_i(t) = r_i x_i(y) - \lambda_i^{\mathrm{Prot}} y_i, \tag{17}$$

where x_i and y_i are the mRNA and protein concentrations of every gene, respectively, m_i is the maximum transcription rate, r_i the translation rate, λ_i^{RNA} and $\lambda_i^{\mathrm{Prot}}$ are the mRNA and protein degradation rates, respectively. The function $f_i(\cdot)$ is the so-called input function of the ith gene, which determines the relative activation of the gene, modulated by the binding of transcription factors (TFs) to *cis*-regulatory sites; it is approximated using Hill-type terms. The typical molecular noise, originated from random thermal fluctuations and other noisy processes attributable to gene expression, was added in equation system (16) and (17) by GNW through Gaussian and lognormal models (for more details see Reference 24). Then, by exploiting GNW, we are able to create noisy artificial datasets for RIVA assessment.

In silico networks of 100 nodes: We have used the same network topologies provided for the DREAM3 Challenge,[18] in particular the subchallenge called *InSilico Size100*, consisting of five networks of 100 nodes, and used these networks to generate the artificial noisy time-course experiments in GNW. In particular, for each network, time-course evolution of the network nodes expression is simulated in response to a simultaneous random perturbation of the basal transcription rate of 10 genes. The network inference algorithms are fed with an experimental batch composed of 10 perturbations, and the targets of the perturbations are chosen in such a way that each gene is perturbed at least once, to evaluate the performance in ideal conditions where all the dynamics of the network are excited.

In silico *Escherichia coli* network of 1565 nodes: We used the *E. coli* transcriptional regulatory network of 1565 nodes provided along with GNW, to generate a case-study for the inference of a large-scale network. As in the case of 100 nodes networks, each experimental batch comprises several time-series (in this case 39 perturbations) with different perturbations in order to modify all the genes: in particular, each perturbation affects the transcriptional rate of a different set of genes (40 genes for the first 38 perturbations, 45 genes for the last one). The inference algorithm has been run on 20 experimental batches, to evaluate the variability of the performance.

2.5 | In vitro micro-array dataset

Experimental micro-array data of the cell cycle regulatory subnetwork in *Saccharomyces cerevisiae* have been extracted from ArrayExpress, a database of functional genomics data.[25] The gold-standard for assessment of the inferred networks has been derived by the model proposed in Reference 26 for transcriptional regulation of cyclin and cyclin/CDK regulators and in Reference 27, where the main regulatory circuits that drive the gene expression program during the budding yeast cell cycle are considered. The network is composed of 27 genes: 10 genes that encode for TF proteins (ace2, fkh1, swi4, swi5, mbp1, swi6, mcm1, fkh2, ndd1, yox1) and 17 genes that encode for cyclin and cyclin/CDK regulatory proteins (cln1, cln2, cln3, cdc20, clb1, clb2, clb4, clb5, clb6, sic1, far1, spo12, apc1, tem1, gin4, swe1 and whi5). Therefore, the expression profiles for the above 27 genes have been extracted from the following two perturbation experiments stored on ArrayExpress:

- **E-MTAB-1908**: This dataset contains the expression levels of *S. cerevisiae* genes, obtained by using metabolic RNA labeling and comparative dynamic transcriptome analysis. In this experimental batch two subdatasets are available, named "*labeling*" and "*total,*" each composed of two independent replicas (named "A" and "B").[28] Since the data are collected every 5 min for three cell cycles and considering the periodic nature of these specific time-series, we have selected a single time period ranging from 5 to 105 min (21 measurement points). The inference tests have been run on the labeling experiment/replica B data.
- **E-MTAB-643**: This experiment investigated the ability of the yeast *S. cerevisiae* to reprogram its transcriptional profile in response to two different nutrient impulses (glucose and ammonium).[29] In this case, the genetic expression profile converges to a steady state. Hence, we have considered only the initial transient, that is the first 2 h of the experiment. The inference tests have been conducted on the data of the glucose impulse experiment.

Before the analysis, the micro-array probe intensity values have been preprocessed via the RMA normalization algorithm[30] (to this aim, the *affyrma* routine in the Matlab Bioinformatics Toolbox has been used). The raw data for both the experiments are normalized together, in order to render the measurements comparable. For achieving a statistical evaluation of the inference algorithms, we have generated 20 artificial replicas by adding the typical noise present in micro-array data.[31]

2.6 | Assessment and benchmarking

Two other state-of-the-art inference algorithms have been used in order to compare the results of RIVA methods:

- *dynGENIE3*[19] is a semi-parametric model, developed as an extension of GENIE3[32] to deal with time-series data. With respect to the GENIE3, which uses tree-based ensemble methods Random Forests or Extra-Trees, the dynGENIE3 algorithm can explicitly take into account the dependence between the values measured at different time points, by using a discrete-time dynamical model of the network to be inferred (similarly to the approach proposed in the present work).
- *BINGO*[20] is a nonparametric approach, based on a gaussian process dynamical models of gene expression profiles and a statistical sampling of the corresponding model realizations using Markov Chain Monte Carlo techniques.

Moreover, in order to quantify the improvement brought by the RLS technique and IV method, we also compare the RIVA results with those obtained by the other LS-based approaches, that is, the standard LS method, its version with the iterative IV algorithm, denoted with LS-IV, and the RLS technique.

The performance of the different methods has been evaluated by computing

- the AUPR index, defined as the area under the precision (or Positive Predictive Value, PPV) versus recall (or sensitivity (Sn)) curve
- the AUROC index, defined as the area under the Receiver Operating Characteristic (ROC) curve, that is, true positive rate versus false-positive rate.

The above indexes take values in the range from 0 (worst case) to 1 (best achievable performance). The positive predictive value is defined as $\mathrm{PPV} := \frac{\mathrm{TP}}{\mathrm{TP+FP}}$ and measures the reliability of the interactions inferred by the algorithm. The sensitivity is defined as $\mathrm{Sn} := \frac{\mathrm{TP}}{\mathrm{TP+FN}}$ and is the fraction of actually existing interactions (FN:=false negatives) with respect to the total number of inferred interactions (see Reference 33, p. 138). To compute these performance indexes, we do not consider the weight of an edge, but only its existence and direction.

3 | RESULTS

In this section RIVA is first assessed through tests on the artificial networks introduced in Section 2.4; subsequently, it is applied to real experimental data for inferring the cell cycle regulatory subnetwork of *S. Cerevisiae* described in Section 2.5.

In both cases, the performance obtained by RIVA are compared with those obtained by BINGO and dynGENIE3 on the same time-series.

The parameters of BINGO have been set as suggested by the authors: the results were computed by generating a total of 6000 samples and keeping all other parameters at default values. Also for dynGENIE3 we have used the suggested default parameters: random forest like method with a number of trees grown in the ensemble equal to 1000. All the experiments can be easily replicated via the scripts provided in Appendix S1 (see the Code and Data Availability section).

3.1 | Tests on the in silico datasets

The first test consists of reconstructing the topology of five networks of 100 nodes, provided in the DREAM3 Challenge, which are used for the generation of our dataset through GeneNetWeaver (see Methods). In particular, the 100-nodes networks are obtained from subnetworks of the *E. coli* and *S. cerevisiae* transcriptional regulatory networks. For each network, all of the inference methods are fed with the 10 time-series in the experimental batch (each time-series comprises 21 time points and is the result of the direct perturbation of the basal expression of 10 genes). The inference tests are replicated five times, using five different realizations of the experimental batches. Figure 1 shows the performance obtained by the three different techniques. The results have been evaluated by computing the AUPR and AUROC indexes (see Methods for the definition of the performance indexes): each plot shows the results for a given network (*E. coli* 1 or 2, Yeast 1, 2 or 3). Figure 1 shows that RIVA outperforms dynGENIE3 and BINGO in terms of AUROC for all the five networks of 100 nodes, while it appears to be comparable (except for Ecoli 1 network) in terms of AUPR.

The capability of RIVA to infer gene regulatory networks, is next tested on a large in silico network. To this aim, we have chosen the *E. coli* transcriptional regulatory network composed of 1565 nodes, and provided by GNW, in the same way as reported in the Section 2. Since the network is very large and the computational time is prohibitive for BINGO, we reported the performances of RIVA and dynGENIE3. Figure 2 shows the results obtained by the two algorithms. The results of RIVA are obtained by generating 20 noisy replicas of the original experimental batch, each batch consisting of 39 time-series of 51 time points; the time-series are generated by perturbing the expression of different genes, so that every genes is perturbed in one of the 39 time-series. The final reconstructed network is obtained via an additional polling step, which takes into account the inference results on the 20 replicated batches: for each inferred network, we create the edge

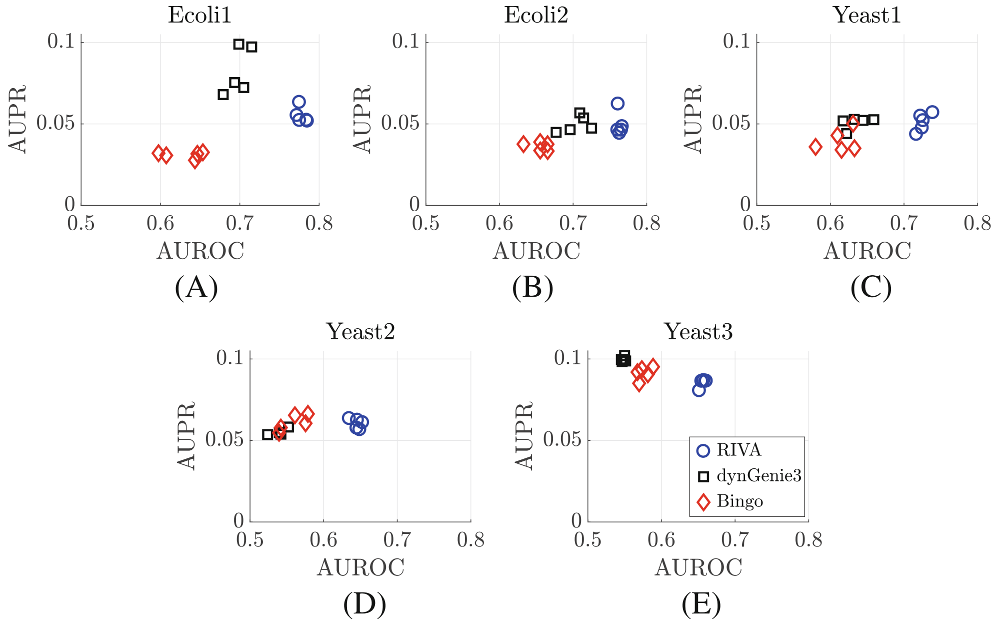

FIGURE 1 Results obtained from dynGenie3 (dark squares), Bingo (red diamonds) and RIVA (blue circles) for the five 100-node networks. The performance (in terms of AUPR and AUROC) for each network is computed over a set of five experimental batches, each one comprising 10 time-series of 21 time points, using a sample time $T_s = 25$ min in the interval [0:500] min

ranking list according to (15); then, the final list of edges is ordered on the basis of the average ranking of each edge over the 20 inference tests. In this case RIVA provides a significant improvement with respect to dynGENIE3 in terms of both AUPR and AUROC, as shown in Figure 2.

To conclude the tests on the artificial networks, we evaluated the performance of RIVA on the dataset provided by the DREAM4 challenge, where time-series are generated by applying a perturbation to some nodes of the network, but only over the first half of the experimental time interval (in the remaining half time interval, the perturbation is removed). In particular, Figure 3 shows the results obtained via RIVA on the DREAM4 networks of 100 nodes. For the evaluation, we generated noisy replicas of the experimental batch (consisting of 10 time-series of 21 points). Furthermore, the influence of the sampling time has also been investigated, by using alternatively $T_s = 50$ min (i.e., equal to that of the original data-set) and $T_s = 25$ min, thereby generating time-series of 41 points; for both the cases ($T_s = 50$ and 25 min) the noisy replicas were obtained by interpolation via smoothing spline, as in Reference 34. For each replica of the experimental batch, we use the first half of the data ([0:500] min) for identifying both the adjacency matrix A and the perturbation target matrix B of system (1) (thus B is considered unknown). Since the perturbations are subsequently removed, in the second half of the experimental time interval ([500:1000] min), only the A matrix is estimated, whereas the B matrix is assumed equal to zero; the results of these two inference steps (with unknown B and with B equal to

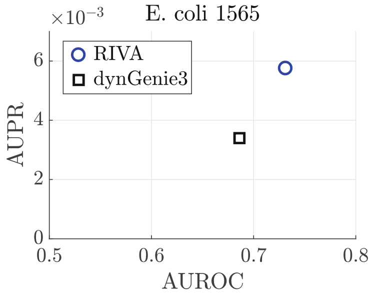

FIGURE 2 Results obtained from dynGenie3 (dark squares) and RIVA (blue circles) on the in silico *E. coli* network of 1565 nodes

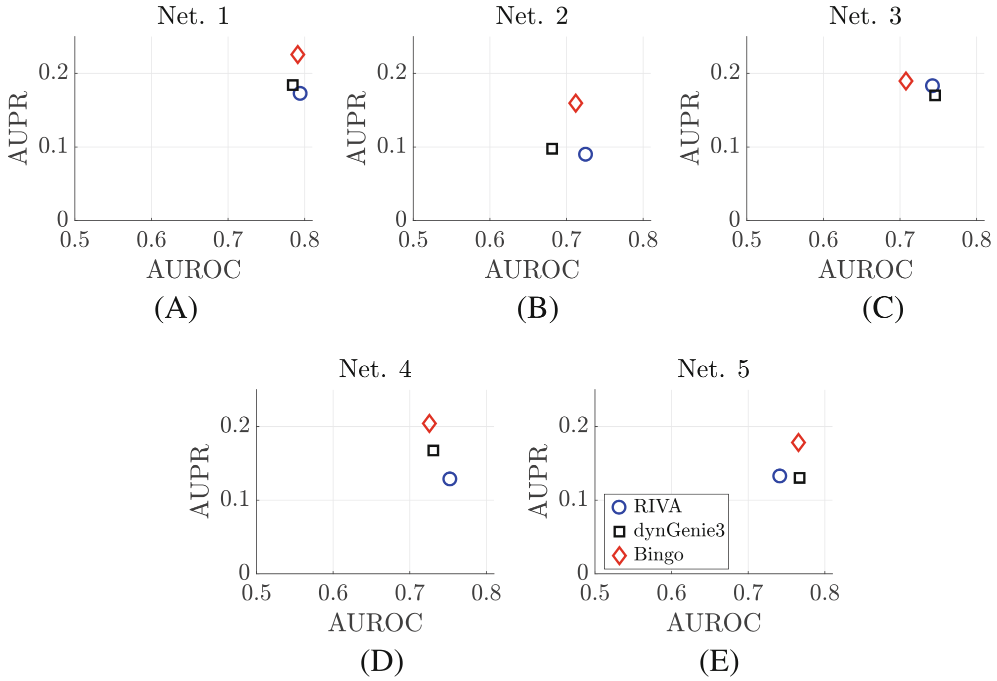

FIGURE 3 Results obtained from dynGenie3 (dark squares), Bingo (red diamonds), and RIVA (blue circles) for the five 100-node networks of DREAM4 challenge, using all the dataset (10 time-series of 21 points), where static perturbations are applied on some nodes for the first half of the data ([0:500] min) and, then, removed for the second half ([500:1000] min)

zero) over the 20 replicated experimental batches are then merged on the basis of the average edge ranking, as illustrated above.

As shown in Figure 3, with this type of dataset BINGO outperforms both dynGENIE3 and RIVA in terms of AUPR (except for Net. 3) and exhibits comparable AUROC values.

As a further test, we have also evaluated the three algorithms using only the first half of the time-series data provided by DREAM4 challenge, that is, without considering the half interval without perturbation. In this case, as shown in Figure 4, RIVA improve considerably, outperforming the other two algorithms. In particular, RIVA shows better results in three of the five analyzed networks (Net.1, Net.3, and Net.5), while, in the other two networks, RIVA shows a slight improvement in terms of AUROC and a slight decrease in terms of AUPR with respect to dynGenie3.

Moreover, for the DREAM4 challenge dataset, we have compared the RIVA results with those obtained via the other LS-based approaches, that is, LS, LS-IV, and RLS, as shown in Appendix S1 (see Figure S1). In almost all cases, RIVA outperforms the other approaches, proving that the combination of the RLS technique and IV method achieves a significantly improved inference capability: indeed, RLS performs better than both LS and LS-IV, and RIVA significantly improves the performance.

3.2 | Assessment with experimental micro-array data

In order to demonstrate the applicability of RIVA to real biological problems, in this section we test its ability to reconstruct a cell cycle regulatory subnetwork in *S. cerevisiae* from experimental micro-array data: the subnetwork consists of 27 genes encoding the main TFs, cyclin, and cyclin/CDK regulatory proteins that determine the essential regulatory circuits driving the gene expression program during the budding yeast cell. For the selected 27 genes, we take the micro-array profiles from ArrayExpress, choosing two datasets, E-MTAB-1908 and E-MTAB-643 (see Section 2). First of all, we have applied RIVA to E-MTAB-643 and E-MTAB-1908 separately. Subsequently, the performance has been evaluated by running RIVA on both the experimental datasets simultaneously, as a single experimental batch. The same type of analysis was performed for dynGENIE3 and BINGO. The results obtained for the three algorithms, on a set of 20 replicated tests, are reported in: (i) Figure 5A–C for E-MTAB-643, (ii) Figure 5D–F for E-MTAB-1908, and (iii) Figure 5G,H for E-MTAB [643:1908]. Each column shows the performance obtained using a given sample time T_s for the two dataset: T_s equal to 15 min for

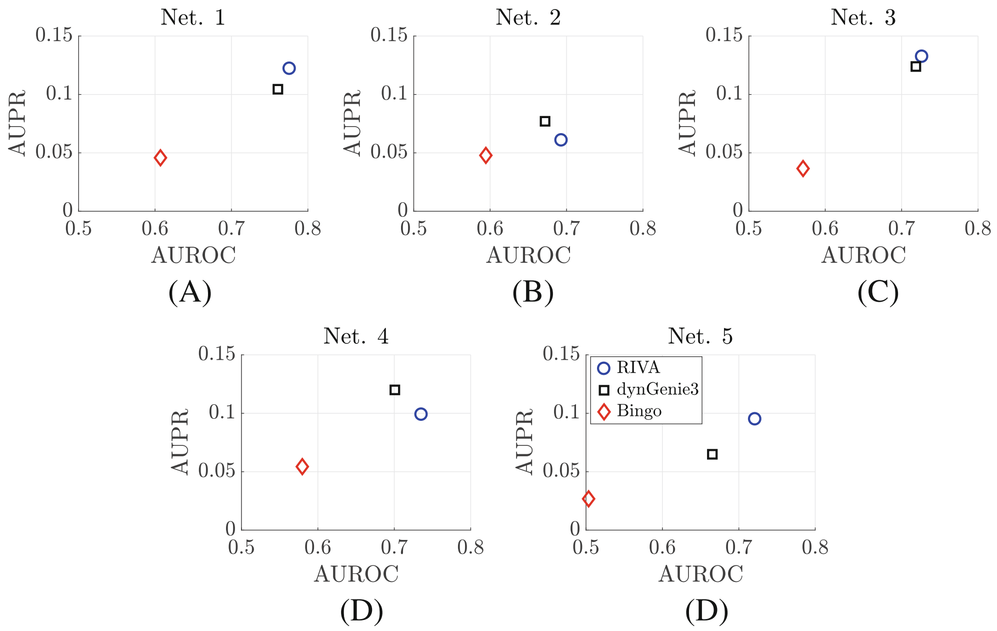

FIGURE 4 Results obtained from dynGenie3 (dark squares), Bingo (red diamonds), and RIVA (blue circles) for the five 100-node networks of DREAM4 challenge, using only the first half of the dataset (10 time-series of 11 points in the interval [0:500] min), where static perturbations are applied on some nodes

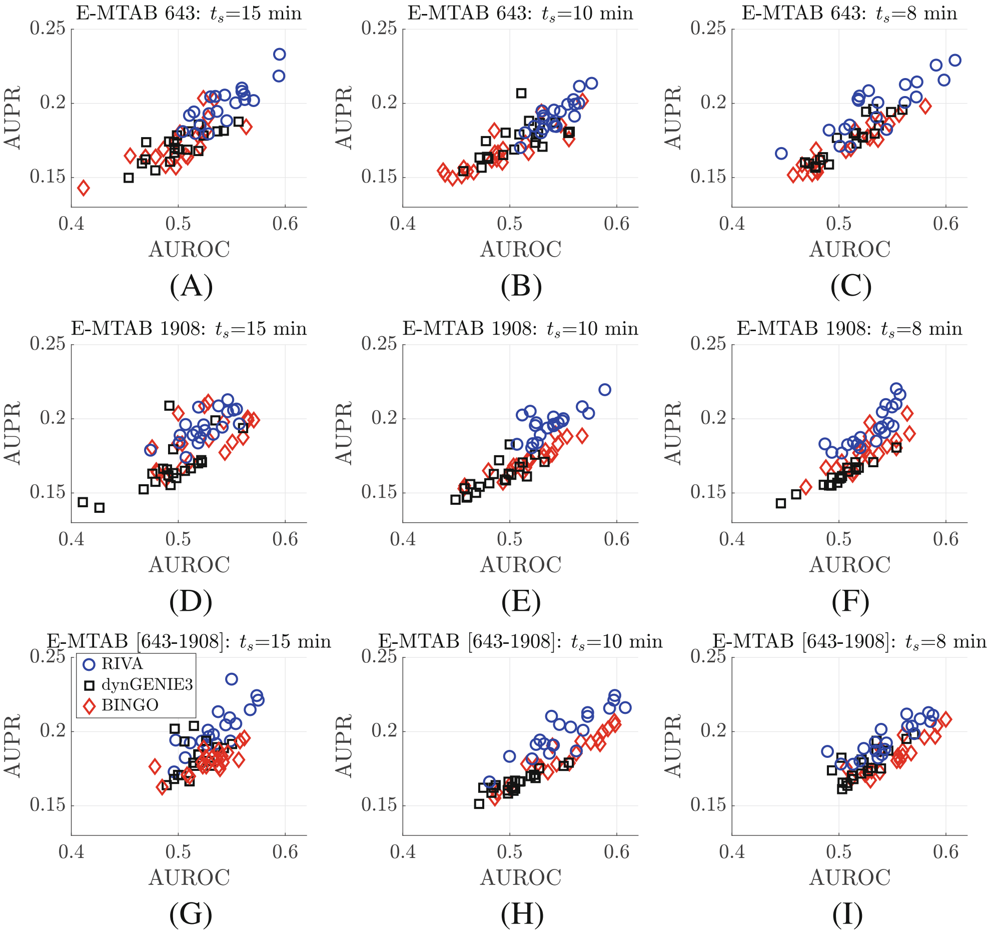

FIGURE 5 Results obtained from dynGenie3 (dark squares), Bingo (red diamonds), and RIVA (blue circles) on the experimental dataset, using different sampling times

the first column, 10 min for the second one and 8 min for the last one. In all of these tests, RIVA appears to perform in a comparable of slightly better way than dynGENIE3 and BINGO: this turns out to be more evident in Figure 6, where the results for the three algorithms have been presented taking into account the mean and the standard deviation. Note that increasing the number of points, that is, decreasing the sample time, does not lead to a significant improvement of RIVA's performance, as well as for dynGENIE3 and BINGO. On the other hand, exploiting simultaneously more than one time-series in the inference process can lead to better performance for RIVA in terms of both AUPR and AUROC (see Figure 6G–I).

3.3 | Computational time cost

With regard to the computational complexity of RIVA, on a standard PC endowed with 2.7 GHz Intel Core i5 processor, a single test on a network of 100 nodes takes about 1 min. In Table 1 we report an estimation of the computational time for networks with different size, to demonstrate that the algorithm can be effectively scaled-up. When the number of nodes is more than 100, the computational time considerably increases due to the estimation of the vector a of the AR filter defined by Equation (9), by using the RLS approach and, in particular, the GCV method for choosing a good estimate of α (see Section 2.2). Regarding the 1565 node network, note that the computational time of RIVA is about 200 times the time required to tackle the 100 node network. Instead, dynGENIE3 requires a computational time about 450 times larger for the large-scale network with respect to the smaller one. If we assume the same scaling factors for BINGO, this would end up to a required computational time approximately between 16 and 37 days to tackle the same large-scale network.

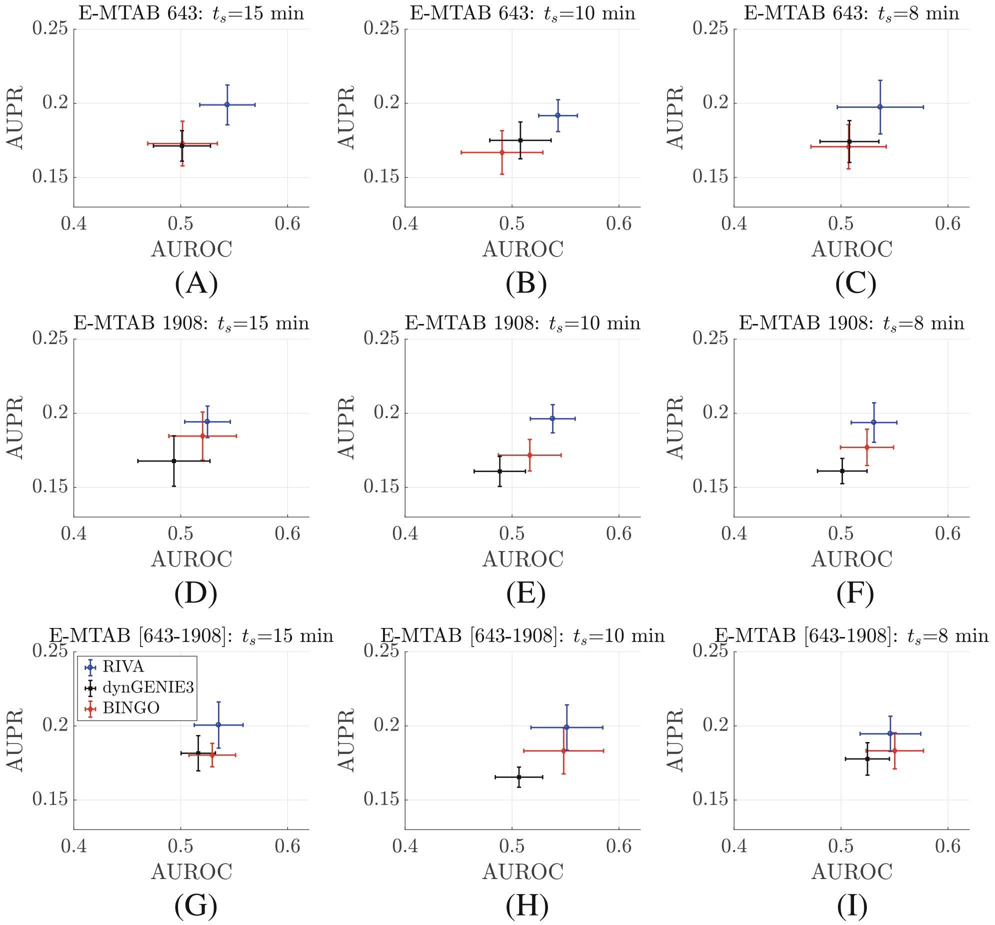

FIGURE 6 Results obtained from dynGenie3 (dark squares), Bingo (red diamonds), and RIVA (blue circles) for the experimental dataset.The results are reported as mean and standard deviation

TABLE 1 Computational time (t_c) (in s) for running RIVA, dynGENIE3, and BINGO on the networks of 100 nodes and of 1565 nodes used in the results section

Algorithm	Net. 100	Net. 1565
RIVA t_c	60	13170
dynGENIE3 t_c	178.71	80578
BINGO t_c	7050	-

4 | CONCLUSIONS

This work has addressed the problem of inferring the topology of a biological network from multiple time-series obtained through perturbation experiments. In order to avoid the issues of bias and nonconsistency of the estimated model parameters, arising in the context of dynamical systems identification due to measurement noise, we have proposed to exploit an algorithm based on the combination of the IV method with a regularization strategy, which is fundamental to deal with the over-parameterization of the regression model. The results obtained on both the artificial networks and the biological case study show that the performance of RIVA is in most cases comparable or better than other state-of-the-art methods. The tests also show that, compared to other approaches in the literature, RIVA is much less computationally demanding. A striking feature of the presented results is the demonstrated ability of an algorithm based on linear models to infer the topology of nonlinear interaction networks. In this paper we have focused our attention on network inference using only time-series data. If additional sources of data, such as steady-state data, are available, then the methods discussed in References 18,35 can also be used to further improve the inference capability.

ACKNOWLEDGMENTS

Francesco Montefusco was supported by the University of Sassari (Fondo di Ateneo per la ricerca 2020). Anna Procopio acknowledges support from Regional Project POR Calabria FESR/FSE 2014-2020.

DATA AVAILABILITY STATEMENT

RIVA has been implemented in the MATLAB software environment; the source code for generating the results presented in the main text is available at https://github.com/BioMecLabUnicz/RIVA. All dataset used in the present work are available at https://mega.nz/folder/StB02LbD#OrU0RVpqYB-P5pU5MC3g4A.

ORCID

Francesco Montefusco https://orcid.org/0000-0002-3264-9686
Carlo Cosentino https://orcid.org/0000-0001-5768-1829

REFERENCES

1. Bansal M, Gatta GD, Di Bernardo D. Inference of gene regulatory networks and compound mode of action from time course gene expression profiles. *Bioinformatics*. 2006;22(7):815-822.
2. Bonneau R, Reiss DJ, Shannon P, et al. The inferelator: an algorithm for learning parsimonious regulatory networks from systems-biology data sets de novo. *Genome Biol*. 2006;7(5):1-16.
3. Yu J, Smith VA, Wang PP, Hartemink AJ, Jarvis ED. Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*. 2009;20:3594-3603.
4. Steuer R, Kurths J, Daub CO, Weise J, Selbig J. The mutual information: detecting and evaluating dependencies between variables. *Bioinformatics*. 2002;18(Suppl. 2):S231-S240.
5. Gardner TS, di Bernardo D, Lorenz D, Collins JJ. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*. 2003;301:102-105.
6. Cosentino C, Curatola W, Montefusco F, Bansal M, di Bernardo D, Amato F. Linear matrix inequalities approach to reconstruction of biological networks. *IET Syst Biol*. 2007;1(3):164-173.
7. Han S, Yoon Y, Cho KH. Inferring biomolecular interaction networks based on convex optimization. *Comput Biol Chem*. 2007;31(5-6):347-354.
8. Montefusco F, Cosentino C, Amato F. CORE–Net: exploiting prior knowledge and preferential attachment to infer biological interaction networks. *IET Syst Biol*. 2010;4(5):296-310.
9. Zavlanos MM, Julius AA, Boyd SP, Pappas GJ. Inferring stable genetic networks from steady-state data. *Automatica*. 2011;47(6):1113-1122.
10. Y. Yuan, G.-B. Stan, S. Warnick, and J. Goncalves, "Robust dynamical network structure reconstruction," *Automatica*, vol. 47, no. 6, pp. 1230–1235, June 2011.
11. Huynh-Thu VA, Sanguinetti G. Combining tree-based and dynamical systems for the inference of gene regulatory networks. *Bioinformatics*. 2015;31(10):1614-1622.
12. Schmidt H, Cho KH, Jacobsen EW. Identification of small scale biochemical networks based on general type system perturbations. *FEBS J*. 2005;272(9):2141-2151.
13. Bansal M, Belcastro V, Ambesi-Impiombato A, di Bernardo D. How to infer gene regulatory networks from expression profiles. *Mol Syst Biol*. 2007;3:1-10.
14. Montefusco F, Cosentino C, Kim J, Amato F, Bates DG. Reverse engineering partially-known interaction networks from noisy data. Proceedings of the 18th IFAC World Congress; August 28 - September 2 2011; Milano, Italy.
15. Nelles O. *Nonlinear System Identification*. Springer-Verlag; 2001.
16. Marbach D, Schaffter T, Mattiussi C, Floreano D. Generating realistic in silico gene networks for performance assessment of reverse engineering methods. *J Comput Biol*. 2009;16(2):229-239.
17. Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, Stolovitzky G. Revealing strengths and weaknesses of methods for gene network inference. *Proc Natl Acad Sci*. 2010;107(14):6286-6291.
18. Prill RJ, Marbach D, Saez-Rodriguez J, et al. Towards a rigorous assessment of systems biology models: the DREAM3 challenges. *PLoS One*. 2010;5(2):e9202.
19. Huynh-Thu VA, Geurts P. dyngenie3: dynamical genie3 for the inference of gene networks from time series expression data. *Sci Rep*. 2018;8(1):1-12.
20. Aalto A, Viitasaari L, Ilmonen P, Mombaerts L, Gonçalves J. Gene regulatory network inference from sparsely sampled noisy data. *Nat Commun*. 2020;11(1):1-9.
21. Kim S, Kim J, Cho KH. Inferring gene regulatory networks from temporal expression profiles under time-delay and noise. *Comput Biol Chem*. 2007;31(4):239-245.

22. Kim J, Bates DG, Postlethwaite I, Heslop-Harrison P, Cho KH. Linear time-varying models can reveal non-linear interactions of biomolecular regulatory networks using multiple time-series data. *Bioinformatics*. 2008;24:1286-1292.
23. Golub GH, Health M, Wahba G. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*. 1979;21(2):215-223.
24. Schaffter T, Marbach D, Floreano D. Genenetweaver: in silico benchmark generation and performance profiling of network inference methods. *Bioinformatics*. 2011;27(16):2263-2270.
25. Parkinson H, Kapushesky M, Shojatalab M, et al. Arrayexpress a public database of microarray experiments and gene expression profiles. *Nucleic Acids Res*. 2006;35(Suppl. 1):D747-D750.
26. Simon I, Barnett J, Hannett N, et al. Serial regulation of transcriptional regulators in the yeast cell cycle. *Cell*. 2001;106(1):697-708.
27. Bähler J. Cell–cycle control of gene expression in budding and fission yeast. *Annu Rev Genet*. 2005;39(1):69-94.
28. Eser P, Demel C, Maier KC, et al. Periodic mRNA synthesis and degradation co-operate during cell cycle gene expression. *Mol Syst Biol*. 2014;10(1):717.
29. Dikicioglu D, Karabekmez E, Rash B, Pir P, Kirdar B, Oliver SG. How yeast re-programmes its transcriptional profile in response to different nutrient impulses. *BMC Syst Biol*. 2011;5(1):148.
30. Irizarry RA, Hobbs B, Collin F, et al. Exploration, normalization, and summaries of high density oligonucleotide array probe level data. *Biostatistics*. 2003;4(2):249-264.
31. Tu Y, Stolovitzky G, Klein U. Quantitative noise analysis for gene expression microarray experiments. *Proc Natl Acad Sci*. 2002;99(22):14 031-14 036.
32. Irrthum A, Wehenkel L, Geurts P, et al. Inferring regulatory networks from expression data using tree-based methods. *PLoS One*. 2010;5(9):e12776.
33. Olson DL, Delen D, eds. *Advanced Data Mining Techniques*. Springer; 2008.
34. Reinsch CH. Smoothing by spline functions. ii. *Numer Math*. 1971;16(5):451-454.
35. Yip KY, Alexander RP, Yan KK, Gerstein M. Improved reconstruction of *in silico* gene regulatory networks by integrating knockout and perturbation data. *PLoS One*. 2010;5(1):e8121.

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section at the end of this article.

How to cite this article: Montefusco F, Procopio A, Bates DG, Amato F, Cosentino C. Scalable reverse-engineering of gene regulatory networks from time-course measurements. *Int J Robust Nonlinear Control*. 2023;33(9):5023-5038. doi: 10.1002/rnc.6044