Cognitive Neurodynamics
https://doi.org/10.1007/s11571-018-9486-0

RESEARCH ARTICLE

PCA based population generation for genetic network optimization

Ahammed Sherief Kizhakkethil Youseph[1] · Madhu Chetty[2] · Gour Karmakar[2]

Received: 22 April 2017 / Revised: 6 February 2018 / Accepted: 24 April 2018

Abstract
A gene regulatory network (GRN) represents a set of genes and its regulatory interactions. The inference of the regulatory interactions between genes is usually carried out using an appropriate mathematical model and the available gene expression profile. Among the various models proposed for GRN inference, our recently proposed Michaelis–Menten based ODE model provides a good trade-off between the computational complexity and biological relevance. This model, like other known GRN models, also uses an evolutionary algorithm for parameter estimation. Considering various issues associated with such population based stochastic optimization approaches (e.g. diversity, premature convergence due to local optima, accuracy, etc.), it becomes important to seed the initial population with good individuals which are closer to the optimal solution. In this paper, we exploit the inherent strength of principal component analysis (PCA) in a novel manner to initialize the population for GRN optimization. The benefit of the proposed method is validated by reconstructing in silico and in vivo networks of various sizes. For the same level of accuracy, the approach with PCA based initialization shows improved convergence speed.

Keywords Principal component analysis (PCA) · Gene regulatory network (GRN) · Computational complexity · Michaelis–Menten kinetics · State space model · Segmentation

Introduction

Cells are the basic structural, functional and biological building blocks of all living organisms. Cellular activities are the results of various biochemical reactions which are catalyzed by enzymes. Proteins, which include all enzymes, are synthesized by ribosomes by decoding the mRNA expressed by genes. Expression of a gene is regulated by several proteins that are essentially gene products. If regulation is such that the presence of a protein activates the expression of a gene, the regulation is called activation. On the other hand, if the regulation is to deactivate the expression of a gene, it is called inhibition. A gene regulatory network (GRN) represents a number of genes interacting in this manner. Gene expression dataset usually comprises the expression levels of the genes as a time-series.

The GRN reconstruction process identifies the regulatory arcs existing between genes from the gene expression data. The key components of the reconstruction process is a model and a method of learning parameters of the model. GRN models proposed in the past range from simple linear models such as Boolean networks (Kauffman 1969) and Bayesian networks (Friedman et al. 2000a, b) to complex nonlinear ordinary differential equation (ODE) models such as S-system (Maki et al. 2001, 2002; Chowdhury et al. 2012). Models with time-delays (Chowdhury et al. 2013; Xiao and Cao 2008; Cao and Ren 2008; He and Cao 2008; Wang et al. 2010; Hu et al. 2013) are also proposed for GRNs.

Among the various models proposed in literature for GRN inference, nonlinear ODE models have been considered better fitting the biological nature of the system. Such models use evolutionary optimization algorithm for the estimation of parameters. Since the initialization of population affects the convergence speed and quality of the optimized network using evolutionary algorithms, several heuristics have been proposed in literature for the

✉ Ahammed Sherief Kizhakkethil Youseph
ahammed.youseph@gmail.com

[1] Faculty of Information Technology, Monash University, Clayton, Australia

[2] Faculty of Science and Technology, Federation University Australia, Gippsland, Australia

Published online: 30 April 2018

improvement of accuracy and acceleration of convergence speed (Ramsey and Grefenstette 1993; Maaranen 2004; Rahnamayan et al. 2007; de Melo and Delbem 2012; Pant et al. 2009; Ali et al. 2013). The generation of a suitable initial population is a major concern in stochastic optimization based GRN reconstruction methods. To address this issue, in this paper, we propose a population initialization method based on principal component analysis of gene expression data. The suitability of the method is demonstrated using Michaelis–Menten decoupled model (MMD-GRN), a nonlinear ODE model for GRN reconstruction proposed in Youseph et al. (2015a). As the parameter estimation of the decoupled model is computationally faster than that of the coupled (MMC-GRN) (Youseph et al. 2015b), we have chosen MMD-GRN model for demonstration.

In computational biology, PCA has already been used for various applications. Some of the key applications, for example, are clustering of genes from the gene expression data (Quackenbush 2001), metabolic profiling by metabolic phenotype clustering from metabolite data (Fiehn et al. 2000), visualizing trajectories of gene expression profiles during different cellular processes (Huang et al. 2005), visualization of gene clusters by compression (Dougherty et al. 2002), studying trends of gene expressions across samples (Riesewijk et al. 2003), etc. Even though the above mentioned applications have vindicated the linearization of a biological system through PCA, however, to the best of our knowledge, no work has so far been carried out in the application of PCA in genetic network reconstruction. There are two issues with regard to using PCA in GRN inference. First, PCA treats the whole dataset as observations obtained from a linear system which is incorrect for any biological system (including GRN) since these systems are highly non-linear. Second, PCA transforms the data into PC-subspace and the output matrices give information on two types of connections: (1) connectivity between the original and transformed variables and (2) between the transformed variables themselves. However, in contrast, in GRN inference, the objective is to identify the regulations of a gene by exploiting the connections between the genes (original variables) themselves.

To address the above mentioned issues, we extend the dynamic application of PCA for GRN modeling, as reported in Bankó et al. (2011), by segmentation of microarray data. The segmentation method ensures that the non-linear biological system can be approximated as linear system, for the short time duration of each segment. To infer the regulations eventually, after applying PCA on each segment, we consider the resulting PC-variables as the states of a state space model, which is a popular linear model in GRN inference. The regulation matrices obtained in each segment are used for generation of the population that is adequate for MMD-GRN model optimization. In brief, we enhance the conventional optimization of the Michaelis–Menten-GRN model by initialization of its population based on PCA.

The effectiveness of the proposed Michaelis–Menten decoupled GRN model with PCA based initialization, referred as MMP-GRN is validated using datasets from the various networks of different sizes. The accuracy of the networks optimized are evaluated in terms of the well-known performance metrics, i.e., sensitivity, specificity, precision and F-score. The speed of inference is measured in terms of simulation time and the number of required function evaluations. The proposed MMP-GRN approach not only produces equal or better accuracy than MMD-GRN, but also shows a significantly reduced computational time.

The rest of the paper is organized as follows. The preliminaries of PCA and state space modeling are described in Sect. 2. The details of the proposed MMP-GRN approach are explained in Sect. 3. Then, the experimental results for MMP-GRN are presented in Sect. 4. Finally, the conclusions are presented in Sect. 5.

Preliminaries

Principal component analysis (PCA)

PCA is one of the widely used statistical methods in dimensionality reduction. PCA carries out a linear transformation of data and projects the data into a much lower-dimensional subspace than the original space.

The classical PCA procedure, considering the whole microarray dataset together is described next. Let $\boldsymbol{X}$ be a dataset, $\boldsymbol{X} \in \mathbb{R}^{N\times T}$ where, N is the number of genes and T is the number of samples.

1. Find $\bar{\boldsymbol{X}}$, the mean-centered data. i.e., calculate the arithmetic mean over each row and deduct the mean value from each element of the row.

$$\bar{\boldsymbol{X}}(i,j) = \boldsymbol{X}(i,j) - \frac{1}{T}\sum_{j=1}^{T}\boldsymbol{X}(i,j)\ \forall\ i = 1,\ldots,N\ ;\ j = 1,\ldots,T$$

2. Find the covariance matrix, $\boldsymbol{\Sigma}_{N\times N} = \frac{1}{T-1}\bar{\boldsymbol{X}}\bar{\boldsymbol{X}}^{\top}$
3. Find the eigen values of $\boldsymbol{\Sigma}$ as $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_M$ where, $M \leq min(N,T)$, is the number of significant eigen values and λ_{M+1} is zero or negligible.
4. Find the eigen vectors corresponding to the above eigen values, $\boldsymbol{e_1}, \boldsymbol{e_2}, \ldots, \boldsymbol{e_M}$ where, $\boldsymbol{e_i} \in \mathbb{R}^{N\times 1}$.

5. $\boldsymbol{\Sigma} = \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^{\top}$ where, $\boldsymbol{W} = [\boldsymbol{e_1}, \boldsymbol{e_2}, \ldots, \boldsymbol{e_M}]$ and $\boldsymbol{\Lambda} = diag(\lambda_1, \lambda_2, \ldots, \lambda_M)$.
6. Each row of $\boldsymbol{W}$ gives the loadings of each gene on the subspace.
7. The observations in the new subspace is given by $\boldsymbol{Z} = \boldsymbol{W}^{\top}\bar{\boldsymbol{X}}$, $\boldsymbol{Z} \in \mathbb{R}^{M \times T}$, the new variables in rows of $\boldsymbol{Z}$ and the observations in its columns.
8. From scores ($\boldsymbol{Z}$) and loadings ($\boldsymbol{W}$), the original data can be reconstructed by $\bar{\boldsymbol{X}} = \boldsymbol{W}\boldsymbol{Z}$ (Fig. 1).

State space modeling

Linear State Space Models (SSM), also called Dynamic Linear Models, are commonly used for modeling dynamical systems. SSM considers the system as being governed by a number of hidden factors, called 'states'. The state space models have two model equations: (1) state and (2) observation equations. Observations are the output or measured variables of the system and states are the hidden factors governing the system. The SSM equations in continuous time are as follows.

The state equation,

$$\dot{\boldsymbol{x}} = \frac{d\boldsymbol{x}}{dt} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \tag{1}$$

and the observation equation,

$$\begin{aligned} &\boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u} \\ &\text{where}, \boldsymbol{x} \in \mathbb{R}^{\mathrm{n}\times 1},\ \boldsymbol{u} \in \mathbb{R}^{\mathrm{p}\times 1},\ \boldsymbol{y} \in \mathbb{R}^{\mathrm{q}\times 1}\boldsymbol{A} \in \mathbb{R}^{\mathrm{n}\times \mathrm{n}}, \\ &\boldsymbol{B} \in \mathbb{R}^{n\times p},\ \boldsymbol{C} \in \mathbb{R}^{q\times n},\ \boldsymbol{D} \in \mathbb{R}^{q\times p} \end{aligned} \tag{2}$$

Here, $\boldsymbol{x}$ is the state vector comprising state variables; $\boldsymbol{u}$ is the input vector consisting of input variables; $\boldsymbol{y}$ is the output vector; n is the number of state variables; p is the number of input variables; q is the number of output variables; $\boldsymbol{A}$ is the state matrix, also called system matrix; $\boldsymbol{B}$ is the input or control matrix; $\boldsymbol{C}$ is the output matrix and $\boldsymbol{D}$ is the feedforward matrix. A generalized model considers all these matrices as time-variants. To account for the system noise and measurement noise, a noise-term can be added in both the equations. In many cases, the system doesn't have a direct feedthrough and $\boldsymbol{D}$ is a zero matrix. No external input to the system is considered in SSM based GRN modeling and hence $\boldsymbol{B}$ is also a zero matrix.

In discrete time, the SSM equations can be expressed as follows.

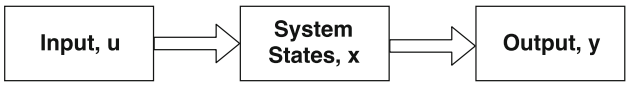

Fig. 1 Illustrtion of the state space modeling framework

$$\boldsymbol{x}_t = \boldsymbol{F}\boldsymbol{x}_{t-1} + \boldsymbol{G}\boldsymbol{u}_{t-1} \tag{3}$$

$$\boldsymbol{y}_t = \boldsymbol{H}\boldsymbol{x}_t + \boldsymbol{D}\boldsymbol{u}_t \tag{4}$$

where, $\boldsymbol{x}_t$ is the state at time t, $\boldsymbol{F}$ is the state transition matrix, $\boldsymbol{H}$ is the observation matrix, $\boldsymbol{u}_t$ and $\boldsymbol{y}_t$ are the input and output at time t, respectively.

The number of states are unknown in the commonly used state-space models for GRN inference. Such models consider different number of states and decide the optimal number of states based on various evaluation criteria. In a GRN model, the gene expression data is considered the output. In a simple SSM with no input and noise, the identification provides the matrix $\boldsymbol{F}$ which indicates the relation between the current state and previous states and the matrix $\boldsymbol{H}$ which indicates the relation between current outputs and current states. However, in GRN inference, the relationship between output variables themselves is to be exploited. A series of matrix operations are proposed in literature to arrive at the GRN structure from the identified matrices (Kojima et al. 2012; Yoshida et al. 2005).

The GRN model

We consider the MMD-GRN model that we proposed in Youseph et al. (2015b). The model equation is as follows.

$$\dot{y_i} = \frac{dy_i}{dt} = \alpha_i \prod_{j=1}^{N} \left(\frac{\frac{\delta_{ij}+1}{|\delta_{ij}|+1} y_j + \frac{1-\delta_{ij}}{|\delta_{ij}|+1} K_{ij}}{K_{ij} + y_j} \right) - \beta_i y_i \tag{5}$$

Here, y_i is the expression level of gene-i, α_i and β_i, respectively are the maximum rate and self decay rate of mRNA expressed by gene-i. δ_{ij} indicates the existence and sign of any regulation from gene-j to gene-i and K_{ij} is the Michaelis–Menten constant corresponding to the regulation, if exists, from gene-j to gene-i.

An ODE based nonlinear model frequently used in GRN modeling is S-system. Parameter estimation of this model is also carried out using stochastic search based optimization algorithms. Hence, we also evaluate the effectiveness of our proposed initialization method on S-system. The model equation is the following.

$$\dot{y}_i = \alpha_i \prod_{j=1}^{N} y_j^{g'_{ij}} - \beta_i \prod_{j=1}^{N} y_j^{h'_{ij}}, \quad i = 1, 2, \ldots, N \tag{6}$$

where, y_i is the expression level of gene-i, N is the total number of genes in the network, α_i and β_i are the kinetic rate constants of production and degradation, respectively of gene-i. The exponents g'_{ij} and h'_{ij} represent the type and strength of the regulation from gene-j to gene-i in production and degradation phases, respectively.

Optimization algorithm

A variant of differential evolution (DE), trigonometric differential evolution (TDE) is applied for estimation of the parameters. In addition to the mutation, cross over and selection operators of DE, a trigonometric mutation operation (TMO) (Fan and Lampinen 2003) is applied in TDE.

PCA based population generation

The data being used for population generation is the time-series gene expression data. The proposed population generation method consists of the following steps: (1) segmentation of the data into locally linear windows of short time-periods, (2) inference of regulations from each segment using PCA and SSM and (3) generation of population adequate for MMD-GRN model optimization from the inferred regulations. The following sections present detailed descriptions of these steps.

Dynamic principal component analysis (DPCA)

The method of PCA assumes the system to be linear and represents the observations as a linear combination of a few number of variables, 'principal components'. However, biological systems are nonlinear in nature. To address this issue, we segment the time-series data into several segments of short time period and carry out the PCA in each segment separately. For a nonlinear dynamic system, the linear assumption is valid if the observations under consideration are for a small time interval. A random segmentation is prone to produce segments violating significantly to the linear approximation. However, a systematic segmentation proposed in Bankó et al. (2011) can ensure that each segment is locally linear or the deviation of the system process in each segment from linearity is negligible.

Segmentation of the time-series data

As biological systems are nolinear and dynamic, the system behavior is not uniform with respect to time. When any disturbance or input is introduced, the system behavior changes abruptly and gradually comes to a steady state. So, the segmentation of time-series data into a fixed number of segments of equal length will be erroneous. It is essential to segment the data by assessing the system behavior in each segment. For this reason, the system response is assessed using a PCA based cost function as in Bankó et al. (2011). The segmentation process is formulated as an optimization problem where the 'cost of merging' is minimized (Bankó et al. 2011). The cost can be based on Hotelling's T^2 statistics or Q reconstruction error whose description is given below:

1. **Hotelling's T^2 statistics**
   T^2 statistic is a measure of the spread of data in the PC-subspace, in other words, variation within the PCA model. Lesser the T^2 value, the lesser the variation of data is. Let $\boldsymbol{Y}_M(k) \in \mathbb{R}^{M\times 1}$ be the observation at k *th* time point for the new variables corresponding to the principal components where, M is the number of principal components considered. The Hotelling's T^2 statistics is defined as:
   $$T^2(k) = \boldsymbol{Y}_M^{\top}(k)\boldsymbol{Y}_M(k) \tag{7}$$
2. **Q reconstruction error (E_Q)**
   Q error or Q statistic is a measure of the fitness of data to the PCA-model or PC-subspace. The lesser the Q error, the better the model fitness. This is the squared Euclidean distance between the original data point and its reconstructed value using only the first Principal Component (PC). Let $\boldsymbol{X}(k) \in \mathbb{R}^{N\times 1}$ be the observations in the original variables at k^{th} time point and $\hat{\boldsymbol{X}}(k)$ be its reconstructed value using the first PC. The Q error (E_Q) is define as:
   $$E_Q(k) = \left(\boldsymbol{X}(k) - \hat{\boldsymbol{X}}(k)\right)^{\top}\left(\boldsymbol{X}(k) - \hat{\boldsymbol{X}}(k)\right) \tag{8}$$

The cost can be formulated as follows.

$$Cost(S_i(p_i, q_i)) = \frac{1}{q_i - p_i + 1}\sum_{k=p_i}^{q_i} l(k) \tag{9}$$

where, S_i is i^{th} segment of the data which spans from the p_i^{th} time sample to the q_i^{th} time sample and $l(k)$ can be either $T^2(k)$ or $E_Q(k)$.

Keogh et al. (2001) grouped the segmentation algorithms into three categories as follows.

1. *Sliding windows* The segmentation algorithm considers the first sample as the first segment and the segment is allowed to grow by adding adjacent samples until a fixed error bound exceeds. Then, the next sample is taken as the next segment and the process is repeated until all samples are processed.
2. *Top-down* The whole time-series data is regarded as one segment. The segmentation algorithm keeps splitting it based on the minimum cost and the process is continued until the defined stopping criterion is met.
3. *Bottom-up* Each sample represents a segment. the costs of merging each pair of adjacent segments are calculated and the segments with the minimum merging cost are merged. This process is continued until the defined stopping criterion is met.

Through the experiments carried out on diverse data-types, Keogh et al. (2001) illustrated that the bottom-up approach is the best suited method for offline segmentation. Therefore, we use the bottom-up approach for the segmentation of our proposed method introduced in this paper. The detailed algorithm for the segmentation method is given in Algorithm 1. The algorithm keeps track of the indices of starting and ending sample of each segment. As bottom-up approach is implemented, the algorithm begins by treating each sample as a separate segment and the costs of merging for each pair of adjacent samples are calculated (Steps 1 and 2). Merging begins with the pair having the lowest merging cost and recalculation of costs and further merging continues until the minimum merging cost exceeds a user-defined threshold, namely *maxerror* (Steps 3 to 20).

Lower the value of *maxerror*, higher the extent of linearity achieved by Algorithm 1. Therefore, *maxerror* makes a trade-off between the number of segments and the extent of linearity achieved by Algorithm 1.

Algorithm 1 Segmentation of the time-series data: - bottom-up approach

Require: $\boldsymbol{X} \in \mathbb{R}^{N \times T}$, the time-series data of N genes and T samples.

Ensure: $\mathbf{a}$, $\mathbf{b}$: vectors of size, N_S, for the indices of the starting and ending sample of segments, respectively. ▷ N_S is the total number of segments obtained.

1: Consider each sample as a separate segment, i.e., $N_S = T$, $\mathbf{a} = [1\ 2\ \ldots\ T]$ and $\mathbf{b} = [1\ 2\ \ldots\ T]$.
2: Using the equation defined in (9), calculate $\mathbf{Cost}(i)$, the cost of merging of i^{th} segment with its adjacent $(i+1)^{th}$ segment, $\forall i = 1, 2, \ldots, (T-1)$.
3: **while** $min(\mathbf{Cost}) \leq maxerror$ **do** ▷ *maxerror* is a user-defined threshold
4: Find the cheapest pair to merge, $k = \underset{i}{\text{argmin}}\ \mathbf{Cost}(i)$
5: Merge the k^{th} segment with the $(k+1)^{th}$ segment.
6: $\mathbf{b}(k) \leftarrow \mathbf{b}(k+1)$.
7: **for** $j = (k+1)$ to $(N_S - 1)$ **do**
8: $\mathbf{a}(j) \leftarrow \mathbf{a}(j+1)$.
9: $\mathbf{b}(j) \leftarrow \mathbf{b}(j+1)$.
10: **end for**
11: Recalculate $\mathbf{Cost}(k-1)$, the merging cost of $(k-1)^{th}$ segment with the k^{th} segment using the equation defined in (9).
12: Recalculate $\mathbf{Cost}(k)$, the merging cost of k^{th} segment with the $(k+1)^{th}$ segment using the equation defined in (9).
13: **for** $j = (k+1)$ to $(N_S - 1)$ **do**
14: $\mathbf{Cost}(j) \leftarrow \mathbf{Cost}(j+1)$.
15: **end for**
16: $\mathbf{a}(N_S) \leftarrow NULL$.
17: $\mathbf{b}(N_S) \leftarrow NULL$.
18: $\mathbf{Cost}(N_S) \leftarrow NULL$.
19: $N_S \leftarrow (N_S - 1)$.
20: **end while**
21: **return** $\mathbf{a}$, $\mathbf{b}$

Inference of genetic regulations

Once the segmentation is carried out, we apply PCA in each segment of the data. PCA provides the scores and the loadings after transformation. We propose to use the principal components as the states of a state space model. The scores of the PCA (See $\boldsymbol{Z}$ defined in Step 7 of the classical PCA procedure described in Sect. 2.1), the observations in the PC-subspace, provide the values of states at different instances of time. From the state transition matrix and the output matrix, we find the connections between the output variables themselves.

Derivation of regulation matrix

The matrix of all regulatory connections between all genes is derived using PCA and SSM. The segmentation algorithm described in Algorithm 1 divides the time-series data into several segments. PCA is applied in each of the segments. Let us consider, for the simplicity of notations, the whole data as a segment. So, the data in the one and only segment is $\boldsymbol{X}_{N \times T}$, the expression of N genes at T time instances. PCA carried out on the matrix $\boldsymbol{X}$ transforms the data into a space of M dimensions where, M is the number of principal components considered and $M \leq min(N, T)$.

Let the score matrix of PCA be $[\boldsymbol{x}(1)\ \boldsymbol{x}(2) \cdots \boldsymbol{x}(T)]$ where, $\boldsymbol{x}(i)$ is a vector of M dimensions. The new dimensions (PCs) are regarded as the states and as mentioned before, $\boldsymbol{F}$ is the state matrix. It may be noted that in an SSM, states are usually hidden variables and as such the values are unknown. However, here we already have the values of states at all time points under consideration. Now, the state equation can be expressed as:

$$\boldsymbol{x}(t+1) = \boldsymbol{F}\boldsymbol{x}(t),\ t = 1, 2, \cdots, (T-1) \tag{10}$$

The matrix form of the above equation to include all time points is the following:

$$\begin{aligned} &[\boldsymbol{x}(2)\ \boldsymbol{x}(3) \cdots \boldsymbol{x}(T)]_{M \times (T-1)} \\ &\quad = \boldsymbol{F}_{M \times M} \times [\boldsymbol{x}(1)\ \boldsymbol{x}(2) \cdots \boldsymbol{x}(T-1)]_{M \times (T-1)} \end{aligned} \tag{11}$$

The observation equation can be expressed using the output matrix $\boldsymbol{H}$ as follows:

$$\boldsymbol{y}(t) = \boldsymbol{H}\boldsymbol{x}(t),\ t = 1, 2, \cdots, (T) \tag{12}$$

where, $\boldsymbol{y}(t)$ is the expression data of N genes at t^{th} time point. Rewriting the equation in matrix form is as follows:

$$\begin{aligned} &[\boldsymbol{y}(1)\ \boldsymbol{y}(2) \cdots \boldsymbol{y}(T)]_{N \times (T)} \\ &\quad = \boldsymbol{H}_{N \times M} \times [\boldsymbol{x}(1)\ \boldsymbol{x}(2) \cdots \boldsymbol{x}(T))]_{M \times T} \end{aligned} \tag{13}$$

Defining new matrices for clarity of derivations:

$$\begin{aligned}
\boldsymbol{X1} &= [\boldsymbol{x}(1)\ \boldsymbol{x}(2)\cdots\boldsymbol{x}(T-1)]_{M\times(T-1)} \\
\boldsymbol{X2} &= [\boldsymbol{x}(2)\ \boldsymbol{x}(3)\cdots\boldsymbol{x}(T)]_{M\times(T-1)} \\
\boldsymbol{X3} &= [\boldsymbol{x}(1)\ \boldsymbol{x}(2)\cdots\boldsymbol{x}(T)]_{M\times(T)} \\
\boldsymbol{Y1} &= [\boldsymbol{y}(1)\ \boldsymbol{y}(2)\cdots\boldsymbol{y}(T-1)]_{N\times(T-1)} \\
\boldsymbol{Y2} &= [\boldsymbol{y}(2)\ \boldsymbol{y}(3)\cdots\boldsymbol{y}(T)]_{N\times(T-1)} \\
\boldsymbol{Y3} &= [\boldsymbol{y}(1)\ \boldsymbol{y}(2)\cdots\boldsymbol{y}(T)]_{N\times(T)}
\end{aligned} \tag{14}$$

Rewriting the state equation [Eq. (11)] using the above defined matrices:

$$\boldsymbol{X2} = \boldsymbol{F}\boldsymbol{X1} \tag{15}$$

System matrix, $\boldsymbol{F}$ can be estimated by solving the above equation in least-squares sense as (Gentle 2007):

$$\boldsymbol{F} = \boldsymbol{X2}\boldsymbol{X1}^{\top}(\boldsymbol{X1}\boldsymbol{X1}^{\top})^{-1} \tag{16}$$

Rewriting the observation equation (Eq. (13)) using the matrices defined in Eq. (14):

$$\boldsymbol{Y3} = \boldsymbol{H}\boldsymbol{X3} \tag{17}$$

The solution for $\boldsymbol{H}$ in least-squares sense is as follows:

$$\boldsymbol{H} = \boldsymbol{Y3}\boldsymbol{X3}^{\top}(\boldsymbol{X3}\boldsymbol{X3}^{\top})^{-1} \tag{18}$$

From Eq. (15), one can understand that the matrix $\boldsymbol{F}$ gives the relation between the state variables themselves. Similarly, by Eq. (17), matrix $\boldsymbol{H}$ gives the relation between the output variables and state variables. However, what is required is the relation between the output variables themselves. For that, we need to find a matrix $\boldsymbol{\Psi}$ where,

$$\boldsymbol{Y2} = \boldsymbol{\Psi}\times\boldsymbol{Y1} \text{ where}, \boldsymbol{Y2} = [\boldsymbol{y}(2)\,\boldsymbol{y}(3)\cdots\boldsymbol{y}(\mathrm{T})]_{\mathrm{N}\times(\mathrm{T}-1)} \tag{19}$$

From Eq. (13),

$$\boldsymbol{y}(2) = \boldsymbol{H}\times\boldsymbol{x}(2) \tag{20}$$

From Eq. (11),

$$\boldsymbol{x}(2) = \boldsymbol{F}\times\boldsymbol{x}(1) \tag{21}$$

Combining the above two equations,

$$\boldsymbol{y}(2) = \boldsymbol{H}\times\boldsymbol{F}\times\boldsymbol{x}(1) \tag{22}$$

In matrix form, the above equation can be rewritten as follows:

$$\boldsymbol{Y2} = \boldsymbol{H}\times\boldsymbol{F}\times\boldsymbol{X1} \tag{23}$$

From Eq. (17),

$$\boldsymbol{Y1} = \boldsymbol{H}\boldsymbol{X1} \Longrightarrow \boldsymbol{X1} = (\boldsymbol{H}^{\top}\boldsymbol{H})^{-1}\boldsymbol{H}^{\top}\boldsymbol{Y1} \tag{24}$$

Therefore,

$$\boldsymbol{Y2} = \boldsymbol{H}\boldsymbol{F}(\boldsymbol{H}^{\top}\boldsymbol{H})^{-1}\boldsymbol{H}^{\top}\boldsymbol{Y1} \tag{25}$$

$$\boldsymbol{Y2} = \boldsymbol{\Psi}\boldsymbol{Y1} \text{ where}, \boldsymbol{\Psi} = \boldsymbol{H}\boldsymbol{F}(\boldsymbol{H}^{\top}\boldsymbol{H})^{-1}\boldsymbol{H}^{\top}. \tag{26}$$

As Eq. (26) relates the current observation vector with the previous observation vector, the elements of the matrix $\boldsymbol{\Psi}$ provide information about the regulations between the genes. A zero value of $\boldsymbol{\Psi}(i,j)$ indicates absence of any regulations from gene-j to gene-i, a positive value of $\boldsymbol{\Psi}(i,j)$ means activation of gene-j by gene-i and a negative value of $\boldsymbol{\Psi}(i,j)$ indicates that gene-i inhibits gene-j. To avoid false regulations, a threshold is set, below which $\boldsymbol{\Psi}(i,j)$ is considered to be zero. The detailed algorithm of inferring regulations using PCA and SSM is given in Algorithm 2.

Algorithm 2 Inferring regulations using PCA and SSM

Require: $\boldsymbol{X}$ - the time-series data of N genes and T samples, a_k, b_k - the indices of the first and last samples, respectively, of k^{th} segment.

Ensure: $\boldsymbol{\Psi}$, the regulation matrix, the sign and magnitude, respectively, of $\boldsymbol{\Psi}(i,j)$ implies the type and strength, respectively, of regulation from gene-j to gene-i.

$\boldsymbol{S} \leftarrow \left[\boldsymbol{X}^{a_k}\ \boldsymbol{X}^{a_k+1}\ \ldots\ \boldsymbol{X}^{b_k}\right]$
Find the mean-centered data, $\bar{\boldsymbol{S}}(i,j) \leftarrow \boldsymbol{S}(i,j) - \frac{1}{L_k}\sum_{j=a_k}^{b_k}\boldsymbol{X}(i,j)\ \forall\ i = 1,2,\ldots,N\ ,\ j = a_k,(a_k+1),\ldots,b_k$. $\triangleright$ $L_k = b_k - a_k + 1$, the length of k^{th} segment.
Find the covariance matrix, $\boldsymbol{\Sigma} = \frac{1}{L_k-1}\bar{\boldsymbol{S}}\bar{\boldsymbol{S}}^{\top}$
Find the eigen values of $\boldsymbol{\Sigma}$ as $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_M$ where, $M \leq min\{N, L_k\}$, λ_{M+1} is zero or negligible
Find the eigen vectors corresponding to the above eigen values, $\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_M$.
$\boldsymbol{W} \leftarrow [\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_M]$.
Find the PCA scores, $\boldsymbol{Z} = \boldsymbol{W}^{\top}\bar{\boldsymbol{S}}$.
Move the PC-axes so that all the scores are non-negative.
for $i = 1$ to M **do**
    $\boldsymbol{Z}_i^{min} \leftarrow min\{\boldsymbol{Z}_i^1,\ \boldsymbol{Z}_i^2\ \ldots,\ \boldsymbol{Z}_i^{L_k}\}$.
    if $\boldsymbol{Z}_i^{min} < 0$ **then**
        for $j = 1$ to L_k **do**
            $\boldsymbol{Z}_i^j \leftarrow \left(\boldsymbol{Z}_i^j - \boldsymbol{Z}_i^{min}\right)$.
        end for
    end if
end for
$\boldsymbol{X1} \leftarrow \left[\boldsymbol{Z}^1\ \boldsymbol{Z}^2\ \ldots\ \boldsymbol{Z}^{L_k-1}\right]$ as defined in Eq. (14).
$\boldsymbol{X2} \leftarrow \left[\boldsymbol{Z}^2\ \boldsymbol{Z}^3\ \ldots\ \boldsymbol{Z}^{L_k}\right]$ as defined in Eq. (14).
$\boldsymbol{X3} \leftarrow \left[\boldsymbol{Z}^1\ \boldsymbol{Z}^2\ \ldots\ \boldsymbol{Z}^{L_k}\right]$ as defined in Eq. (14).
$\boldsymbol{Y1} \leftarrow \left[\boldsymbol{y}^1\ \boldsymbol{y}^2\ \ldots\ \boldsymbol{y}^{L_k-1}\right]$ as defined in Eq. (14).
$\boldsymbol{Y3} \leftarrow \left[\boldsymbol{y}^1\ \boldsymbol{y}^2\ \ldots\ \boldsymbol{y}^{L_k}\right]$ as defined in Eq. (14).
$\boldsymbol{F} \leftarrow \boldsymbol{X2}\times\boldsymbol{X1}^{\top}\times\left(\boldsymbol{X1}\times\boldsymbol{X1}^{\top}\right)^{-1}$ according to Eq. (16).
$\boldsymbol{H} \leftarrow \boldsymbol{Y3}\times\boldsymbol{X3}^{\top}\times\left(\boldsymbol{X3}\times\boldsymbol{X3}^{\top}\right)^{-1}$ according to Eq. (18).
$\boldsymbol{\Psi} \leftarrow \boldsymbol{H}\times\boldsymbol{F}\times(\boldsymbol{H}^{\top}\times\boldsymbol{H})^{-1}\times\boldsymbol{H}^{\top}$ according to Eq. (26).
$\mu_{\boldsymbol{\Psi}} \leftarrow \frac{1}{N\times N}\sum_{i=1}^{N}\sum_{j=1}^{N}|\boldsymbol{\Psi}_{ij}|$.
$\sigma_{\boldsymbol{\Psi}} \leftarrow \sqrt{\frac{1}{N\times N-1}\sum_{i=1}^{N}\sum_{j=1}^{N}(|\boldsymbol{\Psi}_{ij}| - \mu_{\boldsymbol{\Psi}})^2}$.
for $i = 1$ to N **do**
    for $j = 1$ to N **do**
        if $|\boldsymbol{\Psi}_{ij}| < (\mu_{\boldsymbol{\Psi}} - \sigma_{\boldsymbol{\Psi}})$ **then**
            $\boldsymbol{\Psi}_{ij} \leftarrow 0$.
        end if
    end for
end for
return $\boldsymbol{\Psi}$

The PCA is carried out on the segmented data and the PCA scores are calculated (Steps 2 to 7). The PC-axes are moved in the PC-subspace to ensure the scores are non-negative (Steps 9 to 16). The scores are assigned as the state variables of the system and the system and observation matrices are calculated from the states and observations (Steps 17 to 23). The regulation matrix is calculated from the system and observation matrices (Step 24). The mean (μ) and standard deviation (σ) of the absolute regulatory weights are calculated (Steps 25 and 26). Assuming normal distribution of the regulations, 84% of the regulatory weights are expected to be above ($\mu - \sigma$). We have set this value ($\mu - \sigma$) as the threshold below which the regulations are ignored (Steps 27 to 33).

The regulation matrices derived from the segments using PCA and SSM are input to the initial population of the evolutionary optimization algorithm for parameter estimation. For Michaelis–Menten (MM) model that we proposed in Youseph et al. (2015a), namely MMD-GRN defined in Eq. (5), the parameters that decide the regulatory type and strength from gene-j to gene-i are δ_{ij} and K_{ij}, respectively. The regulatory strength is given by the weight function (w_{ij}) that is defined as:

$$w_{ij} = \left(\frac{K_{ij}}{\bar{y}_j}\right)^{\delta_{ij}} \tag{27}$$

where, $\bar{y}_j$ is the average expression level of gene-j. As the value of $\boldsymbol{\Psi}_{ij}$ indicates the strength and type of regulation, if exists, from gene-j to gene-i, the value of $\boldsymbol{\Psi}_{ij}$ can be used to initialize the values for δ_{ij} and K_{ij}. Equating $\boldsymbol{\Psi}_{ij}$ to w_{ij}, the values of δ_{ij} and K_{ij} can be initialized as follows:

$$\delta_{ij} = \begin{cases} 0, \text{ if } \boldsymbol{\Psi}_{\text{ij}} = 0 \\ \dfrac{\boldsymbol{\Psi}_{ij}}{|\boldsymbol{\Psi}_{ij}|}, \text{ otherwise} \end{cases} \tag{28}$$

Since from Eq. (27), $\frac{K_{ij}}{\bar{y}_j} = w_{ij}^{\frac{1}{\delta_{ij}}} = w_{ij}^{\delta_{ij}}$ because of $\delta_{ij} = \pm 1$ for a regulation, we can write K_{ij} as:

$$K_{ij} = \bar{y}_j \times |\boldsymbol{\Psi}_{ij}|^{\delta_{ij}} \tag{29}$$

The detailed procedure of initialization after deriving the regulation matrix, $\boldsymbol{\Psi}$ is shown in Algorithm 3. Using each regulation matrix, N individuals of population to be used for N subproblems of network inference are generated. As presented in equations (28) and (29), for each value of $\boldsymbol{\Psi}_{ij}$, values are assigned for both δ_{ij} and K_{ij} (Steps 1 to 10).

Algorithm 3 Initializing population using regulation matrix, $\boldsymbol{\Psi}$

Require: $\boldsymbol{\Psi} \in \mathbb{R}^{N \times N}$, the regulation matrix and $\bar{y} \in \mathbb{R}^{N \times 1}$, the average levels of expression of N genes.
Ensure: $\boldsymbol{Ind}_1, \boldsymbol{Ind}_2, \ldots, \boldsymbol{Ind}_N$, the set of N individuals initialized, $\boldsymbol{Ind}_i$ being a member of population used for inference of regulations of gene-i.

1: **for** $i = 1$ to N **do**
2: **for** $j = 1$ to N **do**
3: **if** $\boldsymbol{\Psi}_{ij} = 0$ **then**
4: $\delta_j(\boldsymbol{Ind}_i) \leftarrow 0$ as with Eq. (28).
5: **else**
6: $\delta_j(\boldsymbol{Ind}_i) \leftarrow \frac{\boldsymbol{\Psi}_{ij}}{|\boldsymbol{\Psi}_{ij}|}$ as with Eq. (28).
7: **end if**
8: $K_j(\boldsymbol{Ind}_i) \leftarrow \bar{y}_j \times |\boldsymbol{\Psi}_{ij}|^{\delta_{ij}}$ as with Eq. (29).
9: **end for**
10: **end for**
11: **return** $\boldsymbol{Ind}_1, \boldsymbol{Ind}_2, \ldots, \boldsymbol{Ind}_N$

Computational complexity analysis

For the sake of simplicity, in this section, the term complexity is used to represent computational complexity. For a dataset having N genes and T samples, considering the segment of L_k length, the complexity of mean-centering (Step 2 of Algorithm 2) is $\mathcal{O}(L_k \times N)$. The complexity of finding the covariance matrix (Step 3) is $\mathcal{O}(N^2 \times L_k)$. For carrying out the eigen value decomposition (Steps 4 to 6), the complexity is $\mathcal{O}(N^2 \times M)$ (Demmel et al. 2007). To calculate the scores matrix, $\boldsymbol{Z}$ (Step 7), the complexity is $\mathcal{O}(M \times L_k \times N)$. The complexity of modifying scores to ensure non-negativity (Steps 9 to 16) is $\mathcal{O}(M \times L_k)$. As $\boldsymbol{X1}$ and $\boldsymbol{X2}$ are of dimensions $M \times (L_k - 1)$, the complexity of estimating the system matrix $\boldsymbol{F}$ (Step 22) is $\mathcal{O}(M^2 \times (L_k - 1) + M^2 \times (L_k - 1) + M^3 + M^3)$. The matrices $\boldsymbol{Y3}$ and $\boldsymbol{X3}$ are of dimensions $N \times L_k$ and $M \times L_k$, respectively. Thus for finding the observation matrix $\boldsymbol{H}$ (Step 23), the complexity is $\mathcal{O}(N \times M \times L_k + M^2 \times L_k + M^3 + N \times M^2)$. As $\boldsymbol{H}$ and $\boldsymbol{F}$ are of dimensions $N \times M$ and $M \times M$, respectively, for finding the regulation matrix $\boldsymbol{\Psi}$ (Step 24), the complexity is $\mathcal{O}(N \times M^2 + M^2 \times N + M^3 + N \times M^2 + N^2 \times M)$. Since Ψ is an $N \times N$ square matrix, the complexity of the post-processing of obtained regulation matrix (Steps 25 to 33) is $\mathcal{O}(N^2 + N^2 + N^2)$. The number of significant PCs, M will always be $\leq min(L_k, N)$. Moreover, in real-life problems, the number of samples (T) in the time-series data is much less than the number of genes and for each segment, $L_k \leq T$. Assuming this, therefore, the complexity of Algorithm 2 can be regarded as $\mathcal{O}(L_k N^2)$. The total number of segments is N_S. As the segments are non-overlapping, $\sum_{k=1}^{N_S} L_k = T$. Hence, the overall complexity can be written as $\mathcal{O}(TN^2)$.

For the MMD-GRN model, for a single function evaluation and for a single gene, the complexity is $\mathcal{O}(N^2T)$ (Youseph et al. 2015a). Considering N genes and an average of N_f function evaluations required per gene, the overall complexity is $\mathcal{O}(N_fN^3T)$. Since Algorithm 2 requires $\mathcal{O}(TN^2)$, the overall complexity of MMP-GRN is also $\mathcal{O}(N_fN^3T)$. Therefore, both MMD-GRN and MMP-GRN are of the same complexity, $\mathcal{O}(N_fN^3T)$. However, in MMP-GRN, the number of function evaluations required (N_f) is much less than that in MMD-GRN. This is because the initial population of MMP-GRN consists of near-optimal solutions, which makes it faster than MMD-GRN.

Results

The program was coded in MATLAB version 8 (R2012b) and executed in The Monash Campus Cluster (MCC). It may be noted that the matrices $\boldsymbol{H}^\top\boldsymbol{H}$, $\boldsymbol{X1}^\top\boldsymbol{X1}$ and $\boldsymbol{X2}^\top\boldsymbol{X2}$ could be singular. In such cases, the system can have multiple solutions. One of the popular approaches of finding one particular solution is to use Singular Value Decomposition (SVD). In our analysis, we have used the function *pinv* of MATLAB for obtaining pseudoinverse and thus avoiding multiple solutions. For the purpose of evaluating the PCA-based population initialization, we have considered the widely used performance metrics, i.e., sensitivity, precision and F-score for the whole population generated using the proposed method as shown in Table 1 and have compared the results with those results obtained from a set of randomly generated populations. Random populations were generated by allowing random number of regulations for each gene with a limit of 7, a maximum in-degree, typically considered in the GRN reconstruction domain. GRNs are sparse networks, i.e., the non-regulations are much more than the regulations in any given network. As specificity is the true negative rate, it is generally to be high even for random generations and hence is not compared. Following DREAM4 settings of GeneNetWeaver (Schaffter et al. 2011), a benchmarking software for generating realistic *in-silico* genetic networks, ten datasets of 21 samples each were generated for a ten-gene network. The above mentioned performance metrics are calculated for the initial population generated by the proposed method and a number of random initializations.

Table 1 The comparison of the initial populations for the proposed method with random populations, S_p not included as they are similar for all

Method	S_n	P_r	F
The proposed	0.92	0.16	0.27
Random 1	0.37	0.14	0.20
Random 2	0.34	0.15	0.21
Random 3	0.31	0.13	0.18
Random 4	0.41	0.14	0.21
Random 5	0.38	0.13	0.20

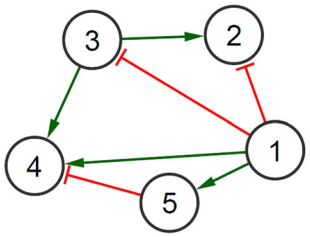

Fig. 2 Five-gene in silico network. Arrow-ended lines indicate activation and bar-ended lines indicate inhibition

Table 2 The MMC-GRN model parameters used in data generation for the five-gene network

Gene (i)	v_i^{max}	v_i^d	K_i	δ_{i1}	δ_{i2}	δ_{i3}	δ_{i4}	δ_{i5}
1	0.40	0.35	0.30	0	0	0	0	0
2	0.50	0.20	–	− 1	0	+ 1	0	0
3	0.30	0.15	0.30	− 1	0	0	0	0
4	0.70	0.50	–	+ 1	0	+ 1	0	− 1
5	0.60	0.40	0.30	+ 1	0	0	0	0

The results are shown in Table 1. The sensitivity for the proposed method is much higher than any random generation, which means that most of the regulations are captured in the proposed initialization. The precision is always higher than any random generation, which means that more of the regulations predicted by the proposed method are true than of those in random generation. As both sensitivity and precision are superior, the harmonic mean of both, F-score is also better for the proposed method than any random generations. The comparison demonstrates that the population generation by the proposed method is always better than any random generation. As the initial population by the proposed method is always of better quality than any random population, the proposed method is expected to show its advantage in the final optimal solution and the convergence rate.

In silico network

Let us first consider a small *in silico* genetic network, shown in Fig. 2, for experiments. This network has five genes and seven regulatory arcs. The gene expression datasets were generated using the Michaelis–Menten model. Following DREAM4 settings of GeneNetWeaver

Table 3 The experimental results (best values) for five-gene synthetic network by MMP-GRN compared with MMD-GRN

Method	S_n	S_p	P_r	F	N_f
MMP-GRN	1.0	1.0	1.0	1.0	16675
MMD-GRN (Youseph et al. 2015b)	1.0	1.0	1.0	1.0	17013
iREGARD-P	1.0	0.74	0.55	0.71	16987
iREGARD (Chowdhury et al. 2014)	0.83	0.79	0.56	0.67	17329

(Schaffter et al. 2011), ten datasets having 21 time samples each were generated. For generation of each dataset, a distinct set of initial conditions (mRNA concentrations) was provided. Needless to mention, the more the number of datasets and samples, the better the inference accuracy will be. The model parameters used for generation of datasets for the five-gene network are tabulated in Table 2.

The results produced using the five-gene network are shown in Table 3. In addition to the well-known performance metrics, i.e. sensitivity, specificity, precision and F-score, the computational speed is assessed in terms of average number of function evaluations required per gene (N_f). The first row in the table shows the results obtained for the proposed method with the Michaelis–Menten model, initialized with populations generated using PCA. The second row shows the results produced by MMD-GRN, the Michaelis–Menten model, with randomly initialized populations.

The same accuracy of 100% is achieved by the MMD-GRN model and MMP-GRN approach. However, the MMP-GRN approach has achieved this with a better computational speed. The average number of function evaluations required per gene (N_f) is reduced by 338 which accounts for approximately 2% reduction. The time required for simulation is 513.9 seconds for MMD-GRN and 498.4 seconds for MMP-GRN. The total simulation time is reduced by 15.5 seconds which account for approximately 3% improvement in time.

In order to evaluate the effectiveness of the proposed initialization method on other existing evolutionary algorithm based GRN inference methods, we have implemented the proposed method on iREGARD (Chowdhury et al. 2014), an S-system based model using differential evolution (DE) algorithm for optimization. S-system model has a different set of parameters, as presented in Sect. 2.3, Eq. (6). In iREGARD, the matrix formed of h_{ij} parameters is assumed to be a diagonal matrix. The same datasets were used for inference. One set of simulations were run using randomly initialized populations and the best values obtained for the performance metrics are presented in the last row of Table 3. Another set of simulations were run using the populations initialized by the proposed method. The regulatory weights obtained using PCA based method are considered as the g_{ij} parameter of the S-system model for initialization. The results obtained are presented in Table 3 where the method is noted as iREGARD-P. As

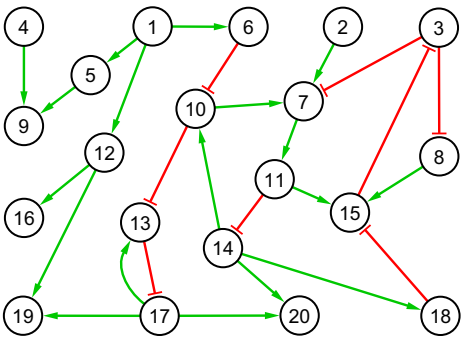

Fig. 3 A twenty-gene synthetic network. Arrow-ended lines indicate activation and bar-ended lines indicate inhibition

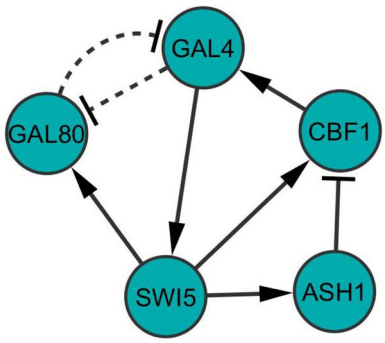

Fig. 4 IRMA Network (Cantone et al. 2009). Solid edges indicate gene interactions and the dashed edges indicates protein-protein interaction. Arrow-ended edges indicate activation and block-ended edges indicate inhibition

Table 4 The experimental results (best values) for 20-gene synthetic network by MMP-GRN compared with MMD-GRN

Method	S_n	S_p	P_r	F	N_f	Simulation time (min)
MMP-GRN	1.0	1.0	1.0	1.0	67989	383.8
MMD-GRN (Youseph et al. 2015b)	1.0	1.0	1.0	1.0	69478	396.9

Table 5 The experimental results for IRMA network, reconstructed from ON and OFF datasets

Method	S_n	S_p	P_r	F	N_f	Simulation time (s)
	ON data					
MMP-GRN	0.71	0.89	0.71	0.71	16965	151.2
MMD-GRN (Youseph et al. 2015b)	0.71	0.89	0.71	0.71	17398	156.1
	OFF data					
MMP-GRN	1.00	0.84	0.67	0.80	17012	152.3
MMD-GRN (Youseph et al. 2015b)	1.00	0.84	0.67	0.80	17215	154.9

observed for Michaelis–Menten model, lesser number of function evaluations is required when initialized with our proposed method compared to a random initialization. The simulation time is also reduced accordingly. Since the datasets were generated using Michaelis–Menten model, the values of the performance metrics are lower for iREGARD compared to Michaelis–Menten model. However, the higher values of sensitivity and F-score is obtained for iREGARD when PCA based population is used in initialization i.e., for iREGARD-P.

A medium-scale synthetic network commonly used by S-system based GRN modelers (Chowdhury 2014; Noman 2007) is considered next and is shown in Fig. 3. The network consists of 20 genes and 22 regulatory arcs.

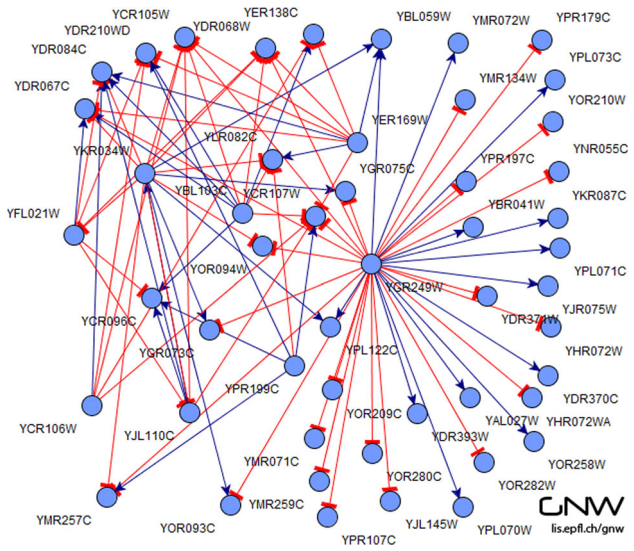

Fig. 5 A 50-gene subnetwork of yeast generated by GeneNetWeaver (Schaffter et al. 2011). Arrow-ended edges indicate activation and block-ended edges indicate inhibition

Michaelis–Menten GRN model is used to generate gene expression data for the network, ten datasets of 21 samples each.

The best of the experimental results obtained are shown in Table 4. Similar to the small network, the inference accuracy is retained, while improving the computational speed. The average number of function evaluations required per gene is reduced by 1489, accounting for approximately 2% reduction. The total simulation time is reduced by 13 minutes which accounts for approximately 3%.

IRMA- an in vivo network

We have also considered the real-life biological data from a subnetwork of yeast, *Saccharomyces cerevisiae*, called

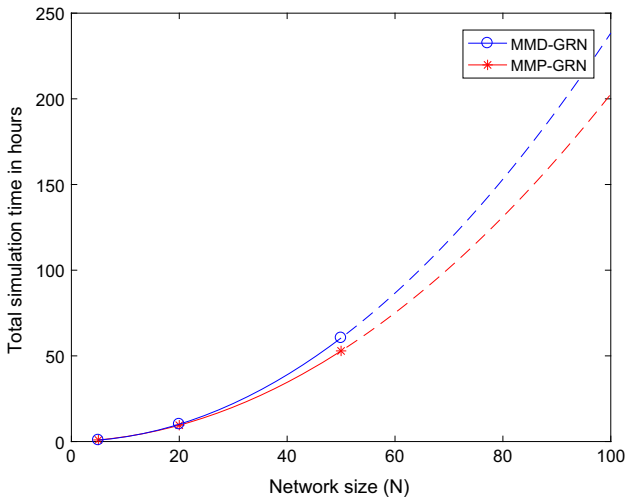

Fig. 6 The average total simulation time per run for genetic networks with varying sizes

Table 6 The best of the experimental results for 50-gene subnetwork of yeast by MMP-GRN compared with MMD-GRN

Method	S_n	S_p	P_r	F	N_f	Simulation time (h)
MMP-GRN	0.46	0.98	0.44	0.45	108762	45.1
MMD-GRN (Youseph et al. 2015b)	0.44	0.98	0.44	0.44	120408	51.3

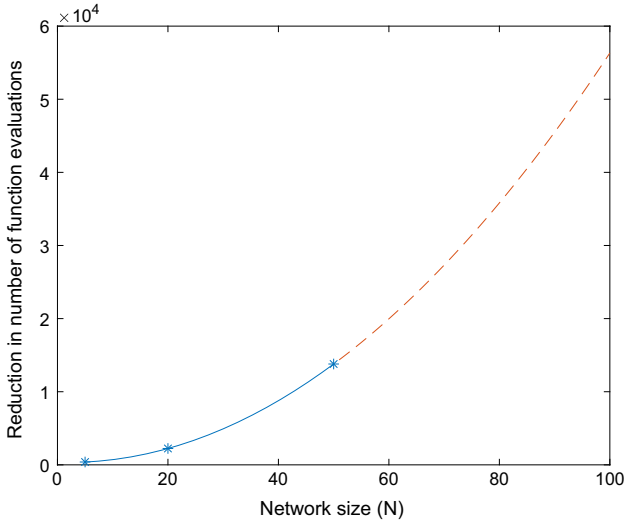

Fig. 7 The reduction in number of function evaluations achieved by MMP-GRN with varying sizes of a GRN

'IRMA' (Cantone et al. 2009), shown in Fig. 4. This is a network of five genes, namely CBF1, GAL4, SWI5, GAL80 and ASH1. *Saccharomyces cerevisiae* can grow on both glucose and galactose; the growth on galactose being more expensive than the growth on glucose. For this reason, the organism always prefers to use glucose for its growth if available (Bhat 2008). However, if glucose is absent in the growth medium, yeast has to look for alternate sources of nutrition, for example, galactose. When galactose is the only available source for growth, yeast synthesizes enzymes that are necessary for the transport as well as metabolism of galactose, i.e., it switches 'ON' the system. Once the organism gets access to glucose, it switches 'OFF' the *GAL* genes. Cantone et al. (2009) generated datasets for both the conditions. (1) Switch 'ON': by growing the yeast *Saccharomyces cerevisiae* in a galactose-rich and glucose-absent medium and (2) switch 'OFF': by transferring the cultures being grown on galactose into a glucose medium. We reconstructed the IRMA network from both the datasets, 'ON' and 'OFF' by the proposed MMP-GRN approach. The experimental results in terms of performance metrics, computational complexity and total simulation time are shown in Table 5. It may be noted that for both ON and OFF data, the proposed MMP-GRN approach reconstructs IRMA network with the same accuracy as MMD-GRN does. However, as expected, the inference by MMP-GRN is achieved in lesser time than MMD-GRN model.

50-gene subnetwork of yeast- an in silico network

We have considered here a subnetwork of Yeast, extracted using GeneNetWeaver. This is a highly sparse network, shown in Fig. 5, having a total of 80 regulatory interactions among the genes. The datasets generated using DREAM4 settings by GeneNetWeaver were used to reconstruct the network using the proposed MMP-GRN approach. The performance metrics for MMP-GRN approach are compared with that for the MMD-GRN (Youseph et al. 2015b) in Table 6.

Although the improvement in accuracy of inference by MMP-GRN over MMD-GRN is limited, we note that the computational complexity is significantly reduced. This is because the average number of function evaluations required per gene is reduced by approximately 10%. The reduction in time for total simulations for a single run is 6.2 hours. In other words, MMP-GRN carries out the inference faster at an approximately 14% higher computational speed, even with the improvement of inference accuracy especially for sensitivity and F-score.

The real-life systems consist of thousands of genes and hence the benefit in terms of speed and accuracy that MMP-GRN bring in will be tremendous for real-world applications. For the time and resource constraints, we have restricted the application of the proposed method to medium scale networks. However, to indicate the benefit of speeding up the algorithm with larger network, we have plotted two graphs: (1) The average total simulation time required versus network size, shown in Fig. 6 and (2) The reduction in number of function evaluations required versus network size (See Fig. 7). The plots were extrapolated upto a network size of 100 using cubic spline method.

The simulation time plot shows that the difference between MMP-GRN and MMD-GRN widens as the network size grows. This implies that the gain in time is huge for large-scale genetic networks. For an unbiased assessment, as an alternate demonstration of the improvement, the reduction in computational complexity in terms of reduction in number of function evaluations required is plotted for varying sizes of GRNs. Similar to the time plot, this plot also shows an exponential decrement of required function evaluations with increase in network size which further validates the advantage of the proposed MMP-GRN approach over MMD-GRN model.

Conclusion

In this paper, we have proposed a knowledge based population generation method using dynamic principal component analysis for genetic network optimization. The superiority of the population generated by the proposed method over any random initializations is demonstrated. The individuals generated for Michaelis–Menten model are used as the initial population of the evolutionary optimization algorithm for GRN inference. The novel MMP-GRN approach is applied for inferring real life and in-silico networks of various sizes. The computational efficiency is assessed in terms of number of function evaluations required and the simulation time. The results show that MMP-GRN approach provides better or equal accuracy and reduced computational complexity for all the networks tested. The reduction in required function evaluations with exponential nature with varying sizes of networks underpins the strength of the proposed method. Eventhough the effectiveness of the proposed initialization method has been demonstrated by embedding into MMD-GRN, it can be used in any GRN inference method with necessary adaptation.

Acknowledgements The authors wish to acknowledge useful suggestions and comments received from the reviewers of the manuscript and from Professor Terry Caelli, The University of Melbourne.

References

Ali M, Pant M, Abraham A (2013) Unconventional initialization methods for differential evolution. Appl Math Compu 219(9):4474–4494. https://doi.org/10.1016/j.amc.2012.10.053. http://www.sciencedirect.com/science/article/pii/S0096300312010697

Bankó Z, Dobos L, Abonyi J (2011) Dynamic principal component analysis in multivariate time-series segmentation. Conserv Inf Evolut 1(1):11–24

Bhat PJ (2008) Galactose regulon of yeast. Springer, Berlin. https://doi.org/10.1007/978-3-540-74015-5

Cantone I, Marucci L, Iorio F, Ricci MA, Belcastro V, Bansal M, Santini S, di Bernardo M, di Bernardo D, Cosma MP (2009) A yeast synthetic network for in vivo assessment of reverse-engineering and modeling approaches. Cell 137(1):172–181. https://doi.org/10.1016/j.cell.2009.01.055

Cao J, Ren F (2008) Exponential stability of discrete-time genetic regulatory networks with delays. IEEE Trans Neural Networks 19(3):520–523. https://doi.org/10.1109/TNN.2007.911748

Chowdhury AR (2014) Gene regulatory network reconstruction using time-delayed S-system model. Ph.D. thesis, Gippsland School of Information Technology, Monash University, Australia

Chowdhury AR, Chetty M, Vinh NX (2012) Adaptive regulatory genes cardinality for reconstructing genetic networks. In: IEEE congress on evolutionary computation (CEC), pp. 1–8. https://doi.org/10.1109/CEC.2012.6256462

Chowdhury AR, Chetty M, Vinh NX (2013) Incorporating time-delays in S-system model for reverse engineering genetic networks. BMC Bioinf 14(1):196. https://doi.org/10.1186/1471-2105-14-196

Chowdhury AR, Chetty M, Vinh NX (2014) Evaluating influence of microRNA in reconstructing gene regulatory networks. Cogn Neurodyn 8(3):251–259. https://doi.org/10.1007/s11571-013-9265-x

de Melo VV, Delbem ACB (2012) Investigating smart sampling as a population initialization method for differential evolution in continuous problems. Inf Sci 193:36–53. https://doi.org/10.1016/j.ins.2011.12.037. http://www.sciencedirect.com/science/article/pii/S0020025512000266

Demmel J, Dumitriu I, Holtz O (2007) Fast linear algebra is stable. Numer Math 108(1):59–91. https://doi.org/10.1007/s00211-007-0114-x

Dougherty E, Barrera J, Brun M, Kim S, Cesar R, Chen Y, Bittner M, Trent J (2002) Inference from clustering with application to gene-expression microarrays. J Comput Biol 9(1):105–126. https://doi.org/10.1089/10665270252833217

Fan HY, Lampinen J (2003) A trigonometric mutation operation to differential evolution. J Global Optim 27(1):105–129. https://doi.org/10.1023/A:1024653025686

Fiehn O, Kopka J, Dormann P, Altmann T, Trethewey RN, Willmitzer L (2000) Metabolite profiling for plant functional genomics. Nat Biotechnol 18(11):1157–1161. https://doi.org/10.1038/81137

Friedman N, Linial M, Nachman I, Pe'er D (2000a) Using Bayesian networks to analyze expression data. J Comput Biol 7(3–4):601–620. https://doi.org/10.1089/106652700750050961

Friedman N, Linial M, Nachman I, Pe'er D (2000b) Using Bayesian networks to analyze expression data. In: Proceedings of the fourth annual international conference on computational molecular biology, RECOMB '00, pp. 127–135. ACM, New York, NY, USA. https://doi.org/10.1145/332306.332355

Gentle JE (2007) Matrix algebra: theory, computations, and applications in statistics. Springer texts in statistics. Springer, New York. https://doi.org/10.1007/978-0-387-70873-7

He W, Cao J (2008) Robust stability of genetic regulatory networks with distributed delay. Cogn Neurodyn 2(4):355. https://doi.org/10.1007/s11571-008-9062-0

Hu J, Liang J, Cao J (2013) Stability analysis for genetic regulatory networks with delays: the continuous-time case and the discrete-time case. Appl Math Comput 220(Supplement C):507 – 517. https://doi.org/10.1016/j.amc.2013.06.003. http://www.sciencedirect.com/science/article/pii/S0096300313006103

Huang S, Eichler G, Bar-Yam Y, Ingber DE (2005) Cell fates as high-dimensional attractor states of a complex gene regulatory network. Phys Rev Lett 94:128,701. https://doi.org/10.1103/PhysRevLett.94.128701

Kauffman SA (1969) Metabolic stability and epigenesis in randomly constructed genetic nets. J Theor Biol 22(3):437–467. https://doi.org/10.1016/0022-5193(69)90015-0. http://www.sciencedirect.com/science/article/pii/0022519369900150

Keogh E, Chu S, Hart D, Pazzani M (2001) An online algorithm for segmenting time series. In: Proceedings 2001 IEEE international conference on data mining, pp. 289–296. https://doi.org/10.1109/ICDM.2001.989531

Kojima K, Imoto S, Yamaguchi R, Fujita A, Yamauchi M, Gotoh N, Miyano S (2012) Identifying regulational alterations in gene regulatory networks by state space representation of vector autoregressive models and variational annealing. BMC Genom 13(Suppl 1):S6. https://doi.org/10.1186/1471-2164-13-S1-S6

Maaranen H, Miettinen K, Kelä MM (2004) Quasi-random initial population for genetic algorithms. Comput Math Appl 47(12):1885–1895. https://doi.org/10.1016/j.camwa.2003.07.011. http://www.sciencedirect.com/science/article/pii/S0898122104840240

Maki Y, Tominaga D, Okamoto M, Watanabe S, Eguchi Y (2001) Development of a system for the inference of large scale genetic networks. In: Pacific symposium on biocomputing vol 6, pp 446–458

Maki Y, Ueda T, Okamoto M, Uematsu N, Inamura K, Uchida K, Takahashi Y, Eguchi Y (2002) Inference of genetic network using the expression profile time course data of mouse P19 cells. Genome Inf 13:382–383. https://doi.org/10.11234/gi1990.13.382

Noman N (2007) A memetic algorithm for reconstructing gene regulatory networks from expression profile. Ph.D. thesis, Graduate School of Frontier Sciences at The University of Tokyo

Pant M, Thangaraj R, Abraha A (2009) Low discrepancy initialized particle swarm optimization for solving constrained optimization problems. Fundam Inf 95(4):511–531. https://doi.org/10.3233/FI-2009-162

Quackenbush J (2001) Computational analysis of microarray data. Nat Rev Genet 2(6):418–427. https://doi.org/10.1038/35076576

Rahnamayan S, Tizhoosh HR, Salama MM (2007) A novel population initialization method for accelerating evolutionary algorithms. Comput Math Appl 53(10):1605–1614. https://doi.org/10.1016/j.camwa.2006.07.013. http://www.sciencedirect.com/science/article/pii/S0898122107001344

Ramsey CL, Grefenstette JJ (1993) Case-based initialization of genetic algorithms. In: Proceedings of the 5th international conference on genetic algorithms, pp 84–91. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. http://dl.acm.org/citation.cfm?id=645513.657423

Riesewijk A, Martín J, van Os R, Horcajadas JA, Polman J, Pellicer A, Mosselman S, Simón C (2003) Gene expression profiling of human endometrial receptivity on days LH+2 versus LH+7 by microarray technology. Mol Hum Reprod 9(5):253–264

Schaffter T, Marbach D, Floreano D (2011) GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods. Bioinformatics 27(16):2263–2270. https://doi.org/10.1093/bioinformatics/btr373

The Monash Campus Cluster. https://confluence.apps.monash.edu/display/MCC/The+Monash+Campus+Cluster. Accessed: 03 April 2017

Wang Y, Cao J, Li L (2010) Global robust power-rate stability of delayed genetic regulatory networks with noise perturbations. Cogn Neurodyn 4(1):81–90. https://doi.org/10.1007/s11571-009-9102-4

Xiao M, Cao J (2008) Genetic oscillation deduced from hopf bifurcation in a genetic regulatory network with delays. Math Biosci 215(1):55–63. https://doi.org/10.1016/j.mbs.2008.05.004. http://www.sciencedirect.com/science/article/pii/S0025556408000783

Yoshida R, Imoto S, Higuchi T (2005) Estimating time-dependent gene networks from time series microarray data by dynamic linear models with Markov switching. In: Proceedings of the IEEE computational systems bioinformatics conference, CSB '05, pp 289–298. IEEE Computer Society, Washington, DC, USA. https://doi.org/10.1109/CSB.2005.32

Youseph ASK, Chetty M, Karmakar G (2015a) Decoupled modeling of gene regulatory networks using Michaelis-Menten kinetics. In: Arik S, Huang T, Lai KW, Liu Q (eds) Neural information processing (Lecture Notes in Computer Science), vol 9491, pp 497–505. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-26555-1_56

Youseph ASK, Chetty M, Karmakar G (2015b) Gene regulatory network inference using Michaelis-Menten kinetics. In: IEEE congress on evolutionary computation (CEC), pp 2392–2397. https://doi.org/10.1109/CEC.2015.7257181