1-1-2007

Inference of Genetic Regulatory Networks with Recurrent Neural Network Models using Particle Swarm Optimization

Rui Xu
Missouri University of Science and Technology

Donald C. Wunsch
Missouri University of Science and Technology, dwunsch@mst.edu

Ronald L. Frank
Missouri University of Science and Technology, rfrank@mst.edu

Recommended Citation

R. Xu et al., "Inference of Genetic Regulatory Networks with Recurrent Neural Network Models using Particle Swarm Optimization," Institute of Electrical and Electronics Engineers (IEEE), Jan 2007.
The definitive version is available at http://dx.doi.org/10.1109/TCBB.2007.1057

Inference of Genetic Regulatory Networks with Recurrent Neural Network Models Using Particle Swarm Optimization

Rui Xu[1], Donald C. Wunsch II[1], and Ronald L. Frank[2]

[1]Applied Computational Intelligence Laboratory, Department of Electrical and Computer Engineering, University of Missouri – Rolla, Rolla, MO 65409-0249 USA

[2]Department of Biological Science, University of Missouri – Rolla, Rolla, MO 65409-0249 USA

***Abstract* – Genetic regulatory network inference is critically important for revealing fundamental cellular processes, investigating gene functions, and understanding their relations. The availability of time series gene expression data makes it possible to investigate the gene activities of whole genomes, rather than those of only a pair of genes or among several genes. However, current computational methods do not sufficiently consider the temporal behavior of this type of data and lack the capability to capture the complex nonlinear system dynamics. We propose a recurrent neural network (RNN) and particle swarm optimization (PSO) approach to infer genetic regulatory networks from time series gene expression data. Under this framework, gene interaction is explained through a connection weight matrix. Based on the fact that the measured time points are limited and the assumption that the genetic networks are usually sparsely connected, we present a PSO-based search algorithm to unveil potential genetic network constructions that fit well with the time series data and explore possible gene interactions. Furthermore, PSO is used to train the RNN and determine the network parameters. Our approach has been applied to both synthetic and real data sets. The results demonstrate that the RNN/PSO can provide meaningful insights in understanding the nonlinear dynamics of the gene expression time series and revealing potential regulatory interactions between genes.**

***Index Terms* —Genetic regulatory networks, Recurrent neural networks, Particle swarm optimization, Time series gene expression data.**

I INTRODUCTION

With the rapid advancement of DNA microarray technologies [10, 29], inferring genetic regulatory networks from time series gene expression data has become critically important to revealing fundamental cellular processes, investigating functions of genes and proteins, and understanding complex relations and interactions between genes [6, 21, 49]. In the context of the data generated from microarray technologies, i.e., transcriptional regulation of protein-coding genes, a genetic regulatory network consists of a set of DNA, RNA, proteins, and other molecules, and it describes regulatory mechanisms among these components. Although

Digital Object Indentifier 10.1109/TCBB.2007.1057

regulation of gene expression can occur at any step along the cellular information flow from DNA to RNA to protein, one of the most common and well-studied steps is the initiation of transcription (RNA synthesis from a DNA template). Eventually, a more complete understanding of gene expression will have to take into consideration that regulation can occur at other levels, e.g., mRNA splicing, translational and post-translational control, including microRNAs that do not encode protein but can interact with DNA or RNA to affect expression. However, the complexity of genetic networks consisting of protein-coding genes that affect other protein-coding genes is vast enough to warrant current studies aimed at data from microarrays. Advancements at this level will facilitate even more complex networks that incorporate proteomic and metabolomic data. All cells for a specific organism include identical genetic information, so it is the regulatory network that determines which subset of genes is expressed, to what level, and in response to what conditions of the cellular environment. For example, genes encoding digestive enzymes are expressed in the gut but not in the skin, and their level of expression increases in the presence of food. This control is achieved through the actions of regulatory proteins, called transcription factors, that activate or inhibit the transcription rate of certain genes by binding to their transcriptional regulatory sites. Therefore, the transcription of a specific gene, or the control of its expression, can be regarded as a combinatorial effect of a set of other genes. In other words, when we say that gene g_1 regulates gene g_2, we actually mean that the transcription factors encoded by g_1, translated from its mRNA products, control the transcription rate of g_2. Other activities, such as RNA splicing and posttranslational modification of proteins, are also constituents of the entire regulatory system. However, due to limited data availability, we can only focus on the transcription (mRNA) level at the current stage instead of the protein level. Despite such restrictions, simplification has the

advantage of measurements over a very large scale, and the use of mRNA expression-based microarrays has led to many interesting and important results [1, 6, 21, 32].

Classical molecular methods, such as Northern blotting, reporter genes, and DNA footprinting, have provided great insight into the regulatory relationships between a pair of genes or among a few preselected genes, which is far from sufficient for exploring their complicated regulatory mechanisms. DNA microarray technologies provide an effective and efficient way to measure the gene expression levels of up to tens of thousands of genes simultaneously under many different conditions; such technologies have already been successfully applied to gene function prediction, disease diagnosis, drug development, and patient survival analysis [30, 48, 50]. Particularly, time series gene expression data, which measure the mRNA abundance of genes through a number of time points, make it possible to investigate gene relations and interactions when taking the entire genome into consideration [6, 21].

Several computational models have been proposed to infer regulatory networks through the analysis of gene expression data [5, 7, 12, 14-17, 20, 23, 28, 31, 33, 36-38, 40-43, 45]. Boolean networks are binary models, which consider that a gene has only two states: 1 for active and 0 for inactive [15, 23, 28, 36]. The effect of other genes on the state change of a given gene is described through a Boolean function. Although Boolean networks make it possible to explore the dynamics of a genetic regulatory system, they ignore the effect of genes at intermediate levels and inevitably cause information loss during the discretization process. Furthermore, Boolean networks assume the transitions between genes' activation states are synchronous, which is biologically implausible. Investigations of the dynamic behaviors under an asynchronous framework are given in [16] and [17]. Bayesian networks are graph models that

estimate complicated multivariate joint probability distributions through local probabilities [12]. Under this framework, a genetic regulatory network is described as a directed acyclic graph that includes a set of vertices and edges. The vertices are related to random variables and are regarded as genes or other components while the edges capture the conditional dependence relation and represent the interactions between genes. Bayesian networks are effective in dealing with noise, incompleteness, and stochastic aspects of gene expression data. However, they do not consider dynamical aspects of gene regulation and leave temporal information unaddressed. Recently, dynamic Bayesian networks (DBN) have attracted more attention [20, 33, 40]. DBN can model behaviors emerging temporally and can effectively handle problems like hidden variables, prior knowledge, and missing data. For linear additive regulation models [5, 7, 37-38], the expression level of a gene at a certain time point can be calculated by the weighted sum of the expression levels of all genes in the network at a previous time point. Although linear additive regulation can reveal certain linear relations in the regulatory systems, it lacks the capability to capture the nonlinear dynamics between gene regulations.

are they in linear combination of the previous time point?

Is RNN the implement ation of the linear additive regulation model?

Considering the limitations of the above methods, we discuss the inference of genetic regulatory networks from time series gene expression in the framework of recurrent neural networks [27]. In using RNNs for genetic network inference, we are mainly concerned with their ability to interpret complex temporal behavior, which is an important characteristic of time series gene expression data and makes them different from static expression data [1]. Generalized RNNs can be considered as signal processing units forming a global regulatory network. The recurrent structure of RNNs effectively reflects the existence of feedback, which is essential for gene regulatory systems. D'haeseleer discussed a realization of RNNs in modeling gene networks using synthetic data [5]. Vohradský investigated the dynamic behaviors of a 3-

gene network in the framework of RNNs [41]. Several more RNN-based applications can also be found in [31], [38], and [45].

Commonly, back-propagation through time (BPTT) [46] and evolutionary algorithms (EAs) [11, 51] are used for RNN training, i.e., learning the functional and structural parameters of regulatory networks. BPTT may be derived by unfolding the temporal operation of the network into a layered feedforward network, the topology of which grows by one layer at every time step [18]. By using BPTT, we find the derivatives of a cost function with respect to the individual weight of the network. These derivatives can be used to do gradient descent on the weights, updating them in the direction that minimizes the error. However, the requirement that the derivatives must be computed limits its application because the derivatives are not always available. EAs are inspired by the process and principles of natural evolution and refer to a class of population-based stochastic optimization search algorithms [51]. The major technologies of EAs include genetic algorithms (GAs), evolution strategies (ES), and evolutionary programming (EP), each of which focuses on a different facet of natural evolution [11, 51]. Particularly, gene regulatory network inference with GAs has already been reported by Wahde and Hertz [42-43] and Keedwell and Narayanan [24]. As an example, Wahde and Hertz used GAs to identify the network parameters in inferring gene regulatory interactions during the development of the central nervous system of rats [42-43]. More broadly, a survey on the application of EAs in classifying biological data was offered by Wahde and Szallasi [44].

Here, we use particle swarm optimization, a variant of evolutionary computation technology for global optimization, for RNN training [9, 26]. In contrast to other EAs, PSO has a random velocity associated with each potential solution, which is considered to be flown through the problem space [26]. Similar to the state-of-the-art EAs, PSO is implemented with a memory

mechanism, which can retain the information of previous best solutions that may get lost during the population evolution. PSO has many other desirable characteristics, such as flexibility in balancing global and local searches, computational efficiency for both time and memory, no need for encoding, and ease of implementation. PSO is particularly useful in evolving neural networks when many local optima exist, a circumstance in which traditional gradient-based search algorithms get stuck easily [26]. It has been shown that PSO requires less computational cost and can achieve faster convergence than conventional back-propagation in training feedforward neural networks for nonlinear function approximation [13]. Juang [22] and Cai and Wunsch [3] combined PSO with EAs in training RNNs for dynamic plant control and engine data classification, respectively. A comparison of PSO and GA in evolving RNNs was also given by Settles et al. [35]. In addition to RNN weight evolution, we also use PSO for RNN architecture evolution in this study. In other words, we search the gene regulatory network (modeled by RNNs) structures in order to unveil the potential and meaningful ones that fit well with the time series data, and we then investigate the interactions between genes. Since genetic regulatory networks are usually assumed to be sparsely connected, this strategy offsets the effect of the insufficient data points of time series to some extent. PSO proves to be a powerful tool to explore sophisticated problem spaces [9, 26], which makes it one of several alternatives that can be explored for regulatory network inference.

The paper is organized as follows. Section II describes the model for regulatory network inference, together with the RNN training algorithm. In Section III, we show how to use PSO to select the potential network structures. Section IV illustrates applications to both the synthetic data and the real data - the SOS DNA repair system. We conclude the paper in Section V.

II. RECURRENT NEURAL NETWORKS

A. Model

For a continuous time system, the genetic regulation model can be represented through a recurrent neural network formulation [5, 31, 38, 41, 45],

$$\tau_i \frac{de_i}{dt} = f(\sum_{j=1}^{N} w_{ij} e_j + \sum_{k=1}^{K} v_{ik} u_k + \beta_i) - \lambda_i e_i, \qquad (1)$$

what is this time constant? why do we need it?

why do we need to multiply the expression itself with the decay rate?

where e_i is the gene expression level for the i^{th} gene ($1 \le i \le N$, N is the number of genes in the system), $f()$ is a nonlinear function (usually, a sigmoid function is used $f(z) = 1/(1+e^{-z})$), w_{ij} represents the effect of the j^{th} gene on the i^{th} gene ($1 \le i,j \le N$), u_k is the k^{th} ($1 \le k \le K$, K is the number of external variables) external variable, which could represent the externally added chemicals, nutrients, or other exogenous inputs, v_{ik} represents the effect of the k^{th} external variable on the i^{th} gene, τ is the time constant, β is the bias term, and λ is the decay rate parameter. A negative value of w_{ij} represents the inhibition of the j^{th} gene on the i^{th} gene, while a positive value indicates the activation controls. When w_{ij} is zero, there is no influence of the j^{th} gene on the expression change of the i^{th} gene. The effects of other factors can be added into the formula based on the specific situation. Note that this model is a natural extension of the linear additive model in [7, 37] in order to explicitly account for the nonlinear dynamics of the networks. Several applications based on the model in Eq. 1 have been reported in the literature [5, 31, 38, 41, 45].

This model can also be described in a discrete form (for computational convenience, since we only measure at certain time points):

$$\frac{e_i(t+\Delta t) - e_i(t)}{\Delta t} = \frac{1}{\tau_i}\left(f(\sum_{j=1}^{N} w_{ij} e_j(t) + \sum_{k=1}^{K} v_{ik} u_k(t) + \beta_i) - \lambda_i e_i(t) \right), \text{ or,}$$

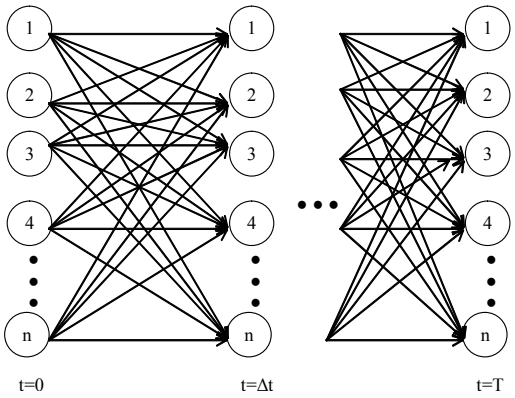


Fig 1. The description of a genetic network through a recurrent neural network model. This network is unfolded in time from t=0 to T with an interval Δt. Here, the regulatory network is shown in a fully connected form, although, in practice, the network is usually sparsely connected.

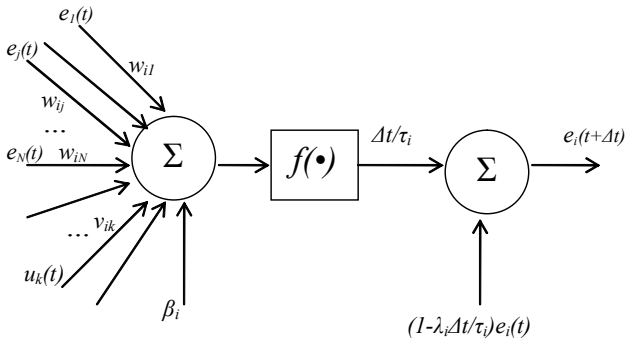


Fig. 2. A node (neuron) in the recurrent neural network model, based on Eq. 2.

$$e_i(t+\Delta t)=\frac{\Delta t}{\tau_i} f\left(\sum_{j=1}^{N} w_{ij} e_j(t)+\sum_{k=1}^{K} v_{ik} u_k(t)+\beta_i\right) + \left(1-\frac{\lambda_i \Delta t}{\tau_i}\right) e_i(t) \quad . \qquad (2)$$

Fig. 1 depicts a recurrent neural network, which is unfolded in time from t=0 to T with an interval Δt, for modeling a genetic network. Here, each node corresponds to a gene, and a connection between two nodes defines their interaction. The weight values can be either positive, negative, or zero, as mentioned above. Fig. 2 illustrates a node in the recurrent neural network, which realizes Eq. 2.

It is usually difficult to obtain the measurements of the external variables, so it is a common practice to ignore the term $\sum_{k=1}^{K} v_{ik} u_k(t)$. Although this simplification inevitably affects the accuracy of the models, studies based on it still provide many interesting insights into gene networks, as demonstrated in [31, 37, 42, 45]. From the following section, we can see that the inclusion of these exogenous inputs does not affect the derivation of the learning algorithm. For computational simplicity, we also assume that the decay rate parameter λ is 1. The final model we process in the paper is represented as

$$e_i(t+\Delta t) = \frac{\Delta t}{\tau_i} \times f\left(\sum_{j=1}^{N} w_{ij} e_j(t) + \beta_i\right) + \left(1 - \frac{\Delta t}{\tau_i}\right) e_i(t). \quad (3)$$

B. Training algorithm

Although the model has already been reported in the literature, the difficulty of RNN training limits its further application for gene network inference, as aforementioned. Here, we propose to use a different training strategy, particle swarm optimization, to determine the unknown network parameters.

PSO consists of a swarm of particles, each of which represents a candidate solution. Each particle i with a position represented as $\mathbf{x}_i$ moves in the multidimensional problem space with a corresponding velocity $\mathbf{v}_i$. The basic idea of PSO is that each particle randomly searches through the problem space by updating itself with its own memory and the social information gathered from other particles. These components are represented in terms of two best locations during the evolution process: one is the particle's own previous best position, recorded as vector $\mathbf{p}_i$, according to the calculated fitness value, and the other is the best position in the whole swarm, represented as $\mathbf{p}_g$. Also, $\mathbf{p}_g$ can be replaced with a local best solution obtained within a certain

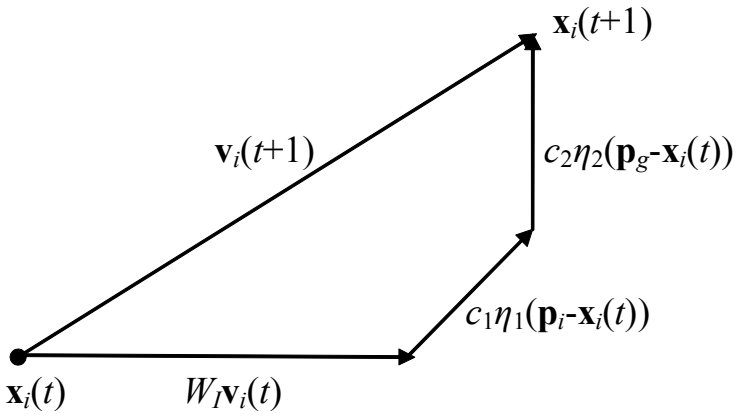


Fig.3. Concept of a swarm particle's position. $\mathbf{x}_i(t)$ and $\mathbf{v}_i(t)$ denote the particle's position and the associated velocity vector in the searching space at generation t, respectively. Vectors $c_1\eta_1(\mathbf{p}_i\text{-}\mathbf{x}_i(t))$ and $c_2\eta_2(\mathbf{p}_g\text{-}\mathbf{x}_i(t))$ describe the particle's cognitive and social activities, respectively. The new velocity $\mathbf{v}_i(t+1)$ is determined by the momentum part, cognitive part, and social part, given in Eq. 4. The particle's position at generation $t+1$ is updated with $\mathbf{x}_i(t)$ and $\mathbf{v}_i(t+1)$, given in Eq. 5.

local topological neighborhood. Fig. 3 depicts the vector representation of the PSO search space. The corresponding canonical PSO velocity and position equations are written as,

$$\begin{aligned}\mathbf{v}_i(t+1) = W_I \times \mathbf{v}_i(t) + c_1 \times \eta_1 \times (\mathbf{p}_i - \mathbf{x}_i(t)) \\ + c_2 \times \eta_2 \times (\mathbf{p}_g - \mathbf{x}_i(t))\end{aligned}, \quad (4)$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \ , \quad (5)$$

where W_I is the inertia weight, c_1 and c_2 are the acceleration constants, and η_1 and η_2 are uniform random functions in the range of [0, 1].

In the context of RNN training with PSO, a set of M particles $\mathbf{X}=(\mathbf{x}_1,\mathbf{x}_2,\ldots,\mathbf{x}_M)$, where the i^{th} particle (candidate solution) can be represented as a D-dimensional vector $\mathbf{x}_i = \left(w_{i,11},...,w_{i,N1},w_{i,12},...,w_{i,1N},...,w_{i,NN},\beta_{i,1},...,\beta_{i,N},\tau_{i,1},...,\tau_{i,N}\right)$ with $D=N(N+2)$, are included in the swarm. The velocity associated with each particle is described as $\mathbf{v}_i = \left(v_{i1},v_{i2},...,v_{iD}\right)$. A fitness function, which is used to measure the deviation of network output $e(t)$ from the real measurement (target) $d(t)$, is defined as

$$Fit(\mathbf{x}_i) = \frac{1}{TN}\sum_{t=0}^{T}\sum_{i=1}^{N}(e_i(t) - d_i(t))^2 . \qquad (6)$$

More elaborate error terms can be added easily based on the specific requirement of the problem at hand. Here, we use a batch mode for training, which means the parameter updates are performed after all input data points are presented to the model [13, 18]. The basic procedure of PSO-based RNN training can be summarized as follows:

i). Initialize a population of particles with random positions and velocities of D dimensions. Specifically, the connection weights, biases, and time constants are randomly generated with uniform probabilities over the range $[w_{\min}, w_{\max}]$, $[\beta_{\min}, \beta_{\max}]$, and $[\tau_{\min}, \tau_{\max}]$, respectively. Similarly, the velocities are randomly generated with uniform probabilities in the range $[-V_{\max}, V_{\max}]$, where $V_{\max}$ is the maximum value of the velocity allowed.

ii). Calculate the estimated gene expression time series based on the RNN model, and evaluate the optimization fitness function for each particle.

iii). Compare the fitness value of each particle $Fit(\mathbf{x}_i)$ with $Fit(\mathbf{p}_i)$. If the current value is better, reset both $Fit(\mathbf{p}_i)$ and $\mathbf{p}_i$ to the current value and location.

iv). Compare the fitness value of each particle $Fit(\mathbf{x}_i)$ with $Fit(\mathbf{p}_g)$. If the current value is better, reset $Fit(\mathbf{p}_g)$ and $\mathbf{p}_g$ to the current value and location.

v). Update the velocity and position of the particles with Eqs. 4 and 5.

vi). Return to step ii until a stopping criterion is met, which usually occurs upon reaching the maximum number of iterations or discovering high-quality solutions.

PSO has only four major user-dependent parameters. The inertia weight W_I is designed as a tradeoff between the global and local search. Larger values of W_I facilitate global exploration while lower values encourage a local search. W_I can be fixed to some certain value or can vary with a random component, such as

$$W_I = W_{\max} - \frac{\eta}{2}, \quad (7)$$

where η is a uniform random function in the range of [0,1]. As an example, if $W_{\max}$ is set as 1, Eq. 7 makes W_I vary between 0.5 and 1, with a mean of 0.75. In this paper, these two strategies are referred to as PSO-FIXEW and PSO-RADW, respectively. c_1 and c_2 are known as the cognitive and social components, respectively, and are used to adjust the velocity of a particle towards $\mathbf{p}_i$ and $\mathbf{p}_g$. Commonly, both parameters are set to 2.0 based on past experience [9, 26]. During the evolutionary procedure, the velocity for each particle is restricted to a limit $V_{\max}$, like in velocity initialization. When the velocity exceeds $V_{\max}$, it is reassigned to $V_{\max}$. If $V_{\max}$ is too small, particles may become trapped into local optima, but if $V_{\max}$ is too large, particles may miss some good solutions. $V_{\max}$ is usually set to around 10-20% of the dynamic range of the variable on each dimension [26].

III. MODEL SELECTION

One of the major obstacles for genetic network inference is the "curse of dimensionality" [5, 38], which describes the exponential growth in computational complexity and the demand for more time points as a result of high dimensionality in the feature space [18]. Typically, the gene expression data currently available contain measurements of thousands of genes, but only with a limited number of time points (less than 50). This situation limits the application of many data-driven computational models and makes it very difficult to infer a fully determined large-scale regulatory network and make accurate predictions of future expression levels. Several strategies have been employed for limiting an effective number of parameters, including clustering [31, 42], interpolation [2, 4-5], adding noisy duplicates [39], and thresholding [38-39]. For RNN training, strategies like weight decays and pruning algorithms can be used [18]. Clustering

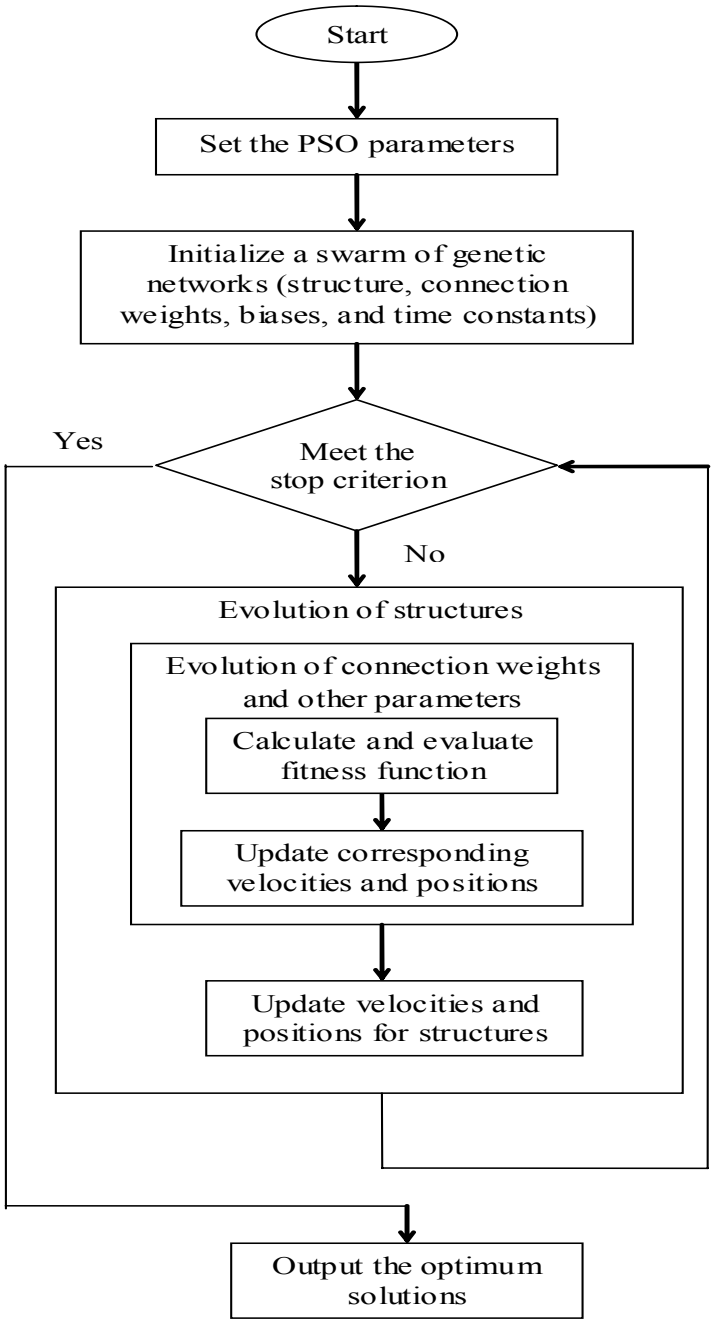


Fig. 4. Flowchart for genetic regulatory network inference with PSO. Both structures and connection weights, together with other parameters, are evolved.

algorithms are used to generate clusters of co-expressed genes based on their expression profiles. For each cluster, the mean time series is then calculated for further analysis. The disadvantage of this method is that it only identifies relations between groups of genes instead of

individual genes, which biologists are more interested in. Interpolation techniques have the disadvantage that they may not sufficiently capture changes between two time points, and they "tend to reduce dimensionality problem only marginally regardless of the number of time points added" [39]. The effectiveness of adding noisy duplicates and thresholding is also limited because these strategies do not provide any additional information on expression level changes.

Fortunately, biological knowledge of genetic regulatory networks assumes that a gene is only regulated by a limited number of genes [5, 33, 38]. In other words, the regulatory networks are sparsely connected rather than fully connected, and most weight values are zeroes. It is reasonable to identify the weights whose values are non-zeroes from these data, which indicate the potential interactions between genes, and furthermore, whether the interaction is activation or inhibition, based on the sign of the weights. However, it may not be possible to accurately recover the values of the weights due to the limited availability of the time points. Wahde and Hertz proposed a two-step procedure for genetic regulatory network inference [43]. The goal of the first step is to unravel the possible interactions between genes by iteratively searching non-significant network parameters, i.e., to determine what weight values are non-zeroes. With the results of the first stage, the non-zero weights can be further fine-tuned, while the non-significant weights are clamped to zero. This procedure is repeated for different values of the maximum allowed weight. This reverse engineering procedure provides a way to identify and understand the regulatory mechanism in a genetic network. However, identifying the non-significant network parameters is not a trivial task, and there is no effective criterion for guidance. A feedforward neural network and genetic algorithm-based hybrid system was proposed in a recent paper by Keedwell and Narayanan [24]. Here, we use PSO to search the network structure space and find meaningful weights that indicate the regulatory relations. This

strategy can avoid the exhaustive enumeration of all possible connectivity (although generally much less than N, the search space is still large), and has the potential to be extended to solve the problem for large-scale regulatory network inference.

Since our goal is to choose a subset of network connections from a large solution space, we employ a discrete binary version of PSO in this context [25]. The major change from the continuous version in Section III lies in the interpretation of the meaning of the particle velocity. As aforementioned, for a set of particles $\mathbf{X}=(\mathbf{x}_1,\mathbf{x}_2,...,\mathbf{x}_M)$, the velocity of the i^{th} particle $\mathbf{x}_i=(x_{i1},x_{i2},...,x_{id})$ is represented as $\mathbf{v}_i^*=(v_{i1}^*,v_{i2}^*,...,v_{id}^*)$, where $d=N^2$. The possible value for each bit x_{il} ($1\le i\le N$, $1\le l\le d$) is either one or zero, which indicates whether there exists an interaction between a pair of genes (1 for yes and 0 for no). The velocity v_{il}^* associated with it is defined as the probability that x_{il} takes the value of one, and it is calculated by the logistic probability law

$$v_{il}(t+1)=W_I\times v_{il}(t)+c_1\times\eta_1\times(p_{i_{il}}-x_{il}(t))+c_2\times\eta_2\times(p_{g_{il}}-x_{il}(t)), \quad (8)$$

$$v_{il}^*(t+1)=1/(1+\exp(-v_{il}(t+1))), \quad (9)$$

$$x_{il}(t+1)=\begin{cases}1 & \text{if } \eta_3+\delta<v_{il}^*(t+1)\\ 0, & \text{otherwise}\end{cases}, \quad (10)$$

where η_3 is a sample of a random variable uniformly distributed in the range of [0, 1], and δ is a parameter that limits the total number of connections selected to a certain range. Compared to the original binary PSO in [25], we add the parameter δ in order to control the connectivity more flexibly. If the value of δ is large, the number of connections for a node becomes smaller, and vice versa. Like the continuous PSO, the velocity is also restricted to a limit $V_{\max}$. In this case,

this restricts the probability that a bit in a particle takes on the value of one to a certain range. Usually, the smaller $V_{\max}$ is, the higher the mutation rate [25].

The inference procedure for the genetic regulatory networks of interest is described through a flowchart in Fig. 4. The algorithm consists of two major steps, i.e., the evolution of the network architecture and the evolution of the corresponding weights, together with other parameters (bias and time constant). For a given network construction, PSO is used to determine the parameters of the genetic networks. A sufficient number of runs are required to assure the quality of the inferred networks. Based on the values of the fitness function and the best networks obtained previously, network structures evolve through another PSO procedure, which aims to explore the meaningful connection relations between network nodes (genes). The procedure iterates until the stop criterion is met. During the algorithm's run, we can also use the strategy introduced by Wahde and Hertz [43] to clamp the structure parameters of significant weights to one when we are highly confident of the presence of gene interactions, or we could clamp the parameters to zero when we are highly confident of the nonexistence of the corresponding connections. This means the entire algorithm may need to be repeated a certain number of times in order to effectively model the regulatory systems.

Since the data are sparse and quite limited, it is meaningless to determine a particular network with the highest fitness value. Rather, potential information could be unveiled by sampling a set of networks and identifying the connection appearing with the highest frequency, as described in a Markov chain Monte Carlo simulation-based method [20]. In our study, we follow the same strategy and determine the network structures based on a large set of sampling networks. In order to estimate the number of connections that a particular network may have, we use a heuristic based on the assumption that the number of times that a nonexistent connection is

observed in the inferred networks follows a binomial distribution with a probability p [47]. We set the level of significance at 0.05, which means the probability of exactly m observations of a nonexistent connection in M sampling networks is less than or equal to 0.05, or $P(m) = C_M^m p^m (1-p)^{M-m} \le 0.05$. We then count the total number of connections NC that occur more than m times, which varies with different values of p. Using this information, we can draw a plot reflecting the relation of NC and p. We select the value of p that corresponds to an abrupt change of the curve as the estimated probability of the false occurrence of a connection. Accordingly, the value of NC, denoted as NC^*, provides an estimate of the real number of connections in the network.

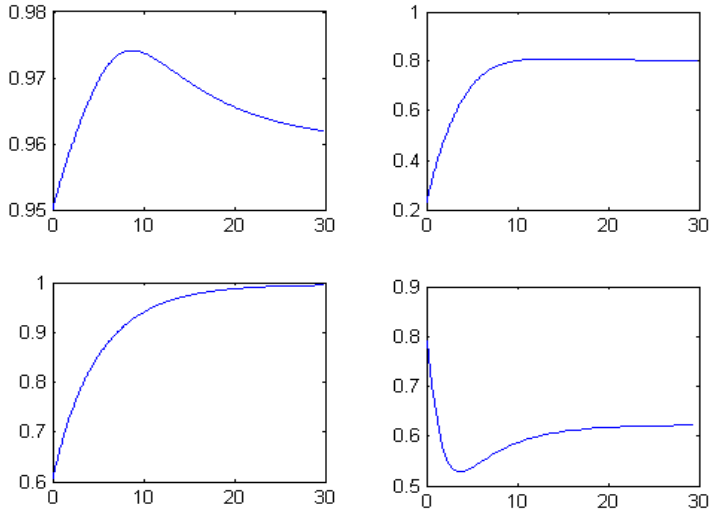


Fig. 5. Typical gene behaviors of the synthetic data in terms of expression level over time. X-axis: time, Y-axis: gene expression level.

Table I. The synthetic genetic network used to generate the data. Half of the weights are non-zeroes.

w_{ij}				β_i	τ_i
20.0	-20.0	0.0	0.0	0.0	10.0
15.0	-10.0	0.0	0.0	-5.0	5.0
0.0	-8.0	12.0	0.0	0.0	5.0
0.0	0.0	8.0	-12.0	0.0	5.0

IV. RESULTS

We perform the proposed PSO/RNN method on both a synthetic and a real data set. By using an artificial data set, we have a clearer view of the performance of the model and algorithm, and we can investigate more effectively the properties of the algorithm as it is possible to compare the learned models with the known regulatory system, which is usually infeasible or incomplete for real data. For the real data set, we show that PSO/RNN can provide meaningful insight into the potential gene interactions in the network.

A. Artificial Data Set

We first apply the algorithm to a simplified synthetic genetic network with 4 genes, used in [42]. The goal is to recover the basic genetic regulatory networks from the generated time series gene expression data. The interaction weight matrix **W**, the bias β, and the time constant τ for the network are set as in Table I. The network is simulated from a random initial state for each gene. We generate three curves with 300 time points for each curve, based on Eq. 3, at a time resolution of Δt=0.1. The typical behaviors of some simulated genes are depicted in Fig. 5. It is clear that, because we do not consider stimuli from the external environment, the expression levels for these genes quickly get saturated. For the real data, the data points generally are not sufficient; therefore, we only use 50 points from each curve to train the regulatory systems, mostly taken from the early stage of the process. For the purpose of comparison, we also perform an experiment for a single time series with 150 time points.

The algorithm is written in C++. It takes about 6 seconds to run 1,000 training epochs with 150 time points on a 2.4GHz Intel Pentium 4 processor with 512M of DDR RAM. Moreover, the code has not been optimized for computational speed. For a large-scale data analysis with

Table II. Parameter effects on PSO performance (average over 200 runs). The parameters are compared in terms of the number of times that the iteration exceeds the allowed maximum and the average number of epochs if PSO has converged.

c_1	c_2	W_I	Performance	
			>500 iterations	Average number of iterations
2	2	Fixed at 0.7	16	53.6
		1-*rand*/2	8	46.9
0.5	2	Fixed at 0.7	104	33.7
		1-*rand*/2	81	46.6
2	0.5	Fixed at 0.7	101	158.0
		1-*rand*/2	54	122.8
0.5	0.5	Fixed at 0.7	180	32.2
		1-*rand*/2	151	18.3
1.5	2.5	Fixed at 0.7	19	56.5
		1-*rand*/2	9	51.1
2.5	1.5	Fixed at 0.7	12	51.5
		1-*rand*/2	7	50.4

hundreds or thousands of genes, parallel technology can be implemented for speedup thanks to the parallel nature of neural networks.

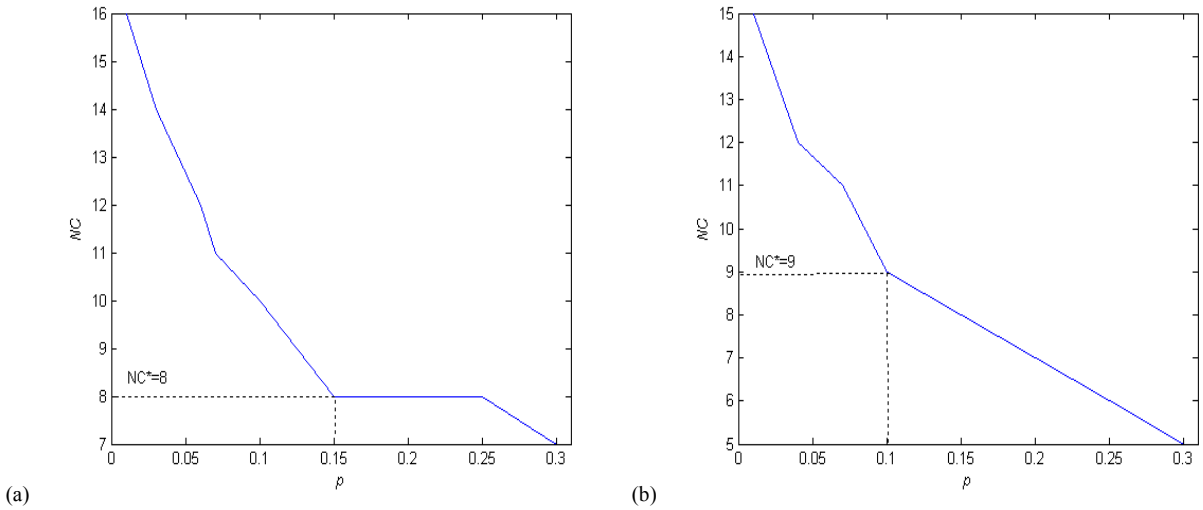
The performance of PSO is dependent on parameter selection. An optimization strategy has been proposed to use another PSO to explore optimal parameter values [8]. However, the computational price may become quite expensive in this situation. We tested the PSO performance with the commonly used values of c_1 and c_2 [8-9], together with both PSO-FIXEW and PSO-RADW methods, on the artificial data set. We fixed the W_I at 0.7 for PSO-FIXEW, and for PSO-RADW, we chose $W_{\max}$ equal to 1 so that W_I varies in the range of [0.5,1]. We compared the performance in terms of the number of iterations required to reach a pre-specified error. We further set the maximum number of iterations allowed as 500. If PSO reaches the expected error threshold within 500 iterations, we say that PSO has converged. The results over 200 runs are summarized in Table II, which consists of the number of times that the iteration exceeds the allowed maximum and the average number of epochs if PSO has converged. As indicated in the table, PSO-RADW performs better than PSO-FIXEW for most of the cases. By further considering the effect of the parameters c_1 and c_2, the best performance is achieved when

PSO-RADW is used and c_1 and c_2 are both set as 2, where the expected error can be reached within 46.9 iterations on average, except for 8 runs that did not converge. The results when c_1 and c_2 are set as 2.5 and 1.5 or 1.5 and 2.5 show that a few more iterations, on average, are required. All three cases are comparable in terms of the number of times that PSO has not converged. For other commonly used parameters, convergence depends more on the initialization and is not consistently stable. Based on the results, we used PSO-RADW with $W_{\max}$ at 1 and set both c_1 and c_2 to 2 in our further experiments.

We performed a random search of the network structure, where a connection between a pair of genes is considered to exist with a probability of 0.5 and the network weights are evolved for 100 iterations using PSO. The average number of iterations for reaching an error level at 10^{-4} using PSO search in 100 runs is 187. This number increases to 1,116 when random search is used in 100 runs. We further decreased the probability of interaction to favor the creation of networks with smaller number of connections. This change achieves certain improvements in the random search method, and the best result we obtained is 892 in 100 runs when the probability is set as 0.3. In another experiment, we added a mechanism in the random search to keep track of all generated structures, and this method generates every structure once, at most. We did not observe significant improvement in this variant. The average number of iterations in 100 runs is 967. We also compared the performance of PSO in network structure search with GA. In this experiment, the RNNs are still trained with PSO, but the network structures are searched using GA. The population size of GA is also 30. A binary tournament selection is used, and the uniform crossover rate is set as 0.8. The mutation probability is chosen at 0.01. The average number of generations of GA in reaching the 10^{-4} error level is 206 in 100 runs, 19 iterations more than PSO. However, we also observe some cases where GA achieves very fast

convergence (within 80 generations). As discussed in [35], PSO and GA have different search capabilities, and the understanding of this difference is important for future research.

We ran our algorithm 200 times with different random initial values for the weights, biases, and time constants. Performance is discussed based on the averages across all these experiments, unless otherwise indicated. The best solution of each run, i.e., $\mathbf{p}_g$, was used for further analysis, which leads to 200 solutions in total. Each swarm consists of 30 particles, with the network structures and the network weights evolved for 100 and 1,000 generations, respectively. We set δ equal to 0.2, which usually leads to the possible connectivity varying between 3 and 12. The initial values for the weights (including biases) and time constants lie between -1 and 1 and 1 and 15, respectively. We estimate the possible number of connections NC^* in the network using the statistical heuristic described in Section III, and we further identify the potential NC^* connections based on the observance frequencies. Moreover, the activation or inhibition relation is determined according to the signs of the mean values of the weights: plus



(a) (b)

Fig. 6. The total number of connections *NC* that occur more than *m* times with the level of significance at 0.05 vs. the probability *p* that a nonexistent connection is observed in a network. (a) Three time series are used; (b) One time series is used.

corresponds to activation while minus represents inhibition. The remaining connections are considered non-existent, with zero weights in the connection weight matrix.

Fig. 6 depicts the changes of *NC* as a function of the probability *p* when three time series or a single time series were used. As we can see, using three time series, when the value of *p* is greater than 0.15, the value of *NC* does not decrease as quickly as when *p* is less than 0.15. Therefore, the estimated value for the possible number of connections NC^* in this network is 8. Similarly, we estimate that there are 9 potential connections in the network when a single time series is used. Table III summarizes the identified weight connection matrices, obtained from either a single time series or multiple time series, together with the original weight matrix.

Table III. The generated connection matrix (upper panel) and the learned connection matrix with the single series (second panel) and multiple series (lower panel). Each element w_{ij} in the matrix represents the relation between the i^{th} and j^{th} gene, as activation (+), inhibition (-), and absence of regulation (0). The values in parentheses indicate the percentage of the occurrence of connections in the networks within 200 runs.

w_{ij} (f%)			
+	-	0	0
+	-	0	0
0	-	+	0
0	0	+	-
0 (11.5)	+ (27.5)	- (25.5)	- (44.0)
0 (3.0)	- (20.0)	- (17.5)	- (52.0)
0 (1.0)	0 (10.0)	0 (4.5)	0 (8.0)
0 (2.5)	+ (55.0)	+ (32.5)	- (59.5)
+ (29.0)	+ (34.0)	0 (5.5)	0 (7.0)
+ (83.5)	- (98.0)	0 (3.5)	0 (3.0)
0 (16.5)	- (60.5)	+ (99.5)	0 (15.0)
0 (10.0)	0 (8.5)	+ (100)	- (100)

Compared with the original weight matrix, the result with three time series demonstrates that the proposed model can recover all eight relations existing in the network without any false positives, which refer to non-existent relations that the model wrongly identifies as existing ones. Particularly, all 200 solutions identify the activation relation of gene 4 to gene 3 and the

self-inhibition of gene 4. The only mistake lies in the inhibition of genes 1 to gene 2, which is identified as activation in the constructed network. In contrast, using a single time series is ineffective, even though the mean square error is small (at the level of 10^{-5}). Only four out of eight non-zero weights are correctly identified, but with five false positives. Also, the identified relation between gene 1 and 2 is wrongly regarded as activation. These results agree with the conclusion in [42-43] that by using more time series, more information is provided to the model, and therefore, better results usually can be achieved.

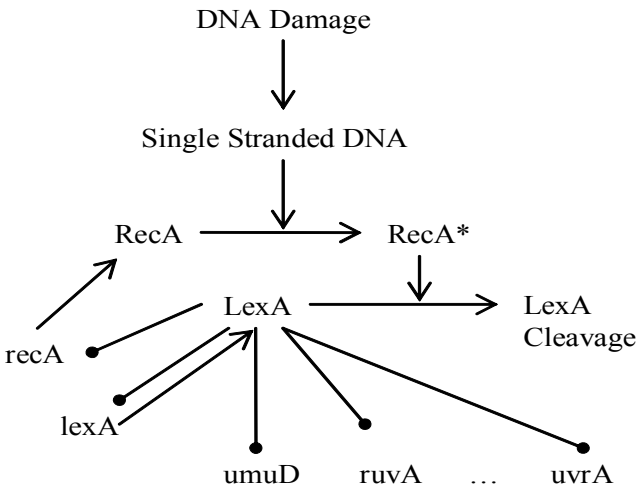


Fig. 7. The bacterial *E. coli* SOS DNA Repair network. Inhibitions are represented by -•, while activations are represented by →. When damage occurs, the protein RecA, which functions as a sensor of DNA damage, becomes activated and mediates cleavage of the LexA protein. The LexA protein is a repressor that blocks transcription of the SOS repair genes. The drop in LexA protein levels causes the activation of the SOS genes. After the damage is repaired or bypassed, the cleavage activity of RecA drops, causing the accumulation of LexA protein. Then, LexA binds sites in the promoter regions of these SOS genes and represses their expression. The cells return to their original states [34].

B. SOS Data Set

We employed PSO/RNN to analyze the SOS DNA Repair network in bacterium *Escherichia coli* depicted in Fig. 7 [34]. The SOS system consists of around 30 genes regulated at the

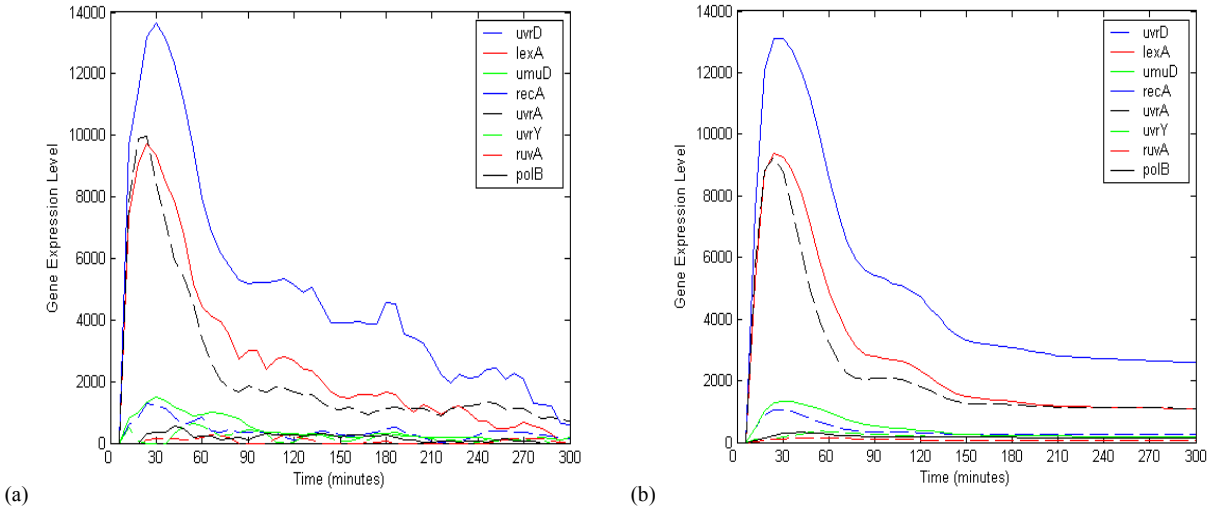


Fig. 8. (a) The measured gene expression profiles for Exp. 2; (b) the learned mean expression profiles with PSO. The average mean square error between the real profiles and learned profiles (both are normalized into the range [0, 1]) is 0.022.

transcriptional level. For the data set, four experiments have been conducted with different light intensities (Experiments 1 & 2: UV=5 Jm^{-2}, Experiments 3 & 4: UV=20 Jm^{-2}). Each experiment includes the expression measurements for 8 major genes (*uvrD*, *lexA*, *umuD*, *recA*, *uvrA*, *uvrY*, *ruvA*, and *polB*) through 50 time points, sampled every 6 minutes. Therefore, the data set consists of 4 subsets, each of which is represented as a matrix $\mathbf{ET}_i$, i=1,…,4. To our knowledge, this is one of the most useful data sets that fits the current computational models, as indicated in [33]. Like before, we performed analyses on both single time series and multiple time series.

Fig. 8 shows the real gene expression profiles and the learned mean expression profiles for Exp. 2. We can see that the proposed model can effectively capture the dynamics of most genes *(lexA*, *recA*, *uvrA*, *uvrD*, and *umuD*) in the system, with the major change trends of the gene expression levels reflected in the learning curves. The expression profiles for genes *uvrY*, *ruvA*, and *polB* oscillate dramatically between the maximum value and zero, and the obtained models generally use their means to represent the profiles because of the definition of the fitness function. One possible strategy is to add an additional item in the fitness function in order to

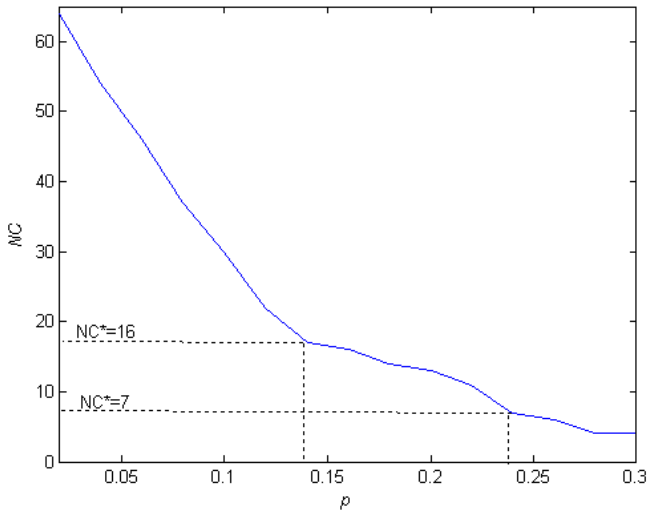


Fig. 9. Estimation of the number of connections in the SOS network.

track the difference between two time points. The reader should note that reconstructing the correct dynamics is easier than reconstructing the correct regulatory network structure because genetic regulatory network inference is an ill-posed problem – there is no unique solution that will satisfy the data upon which the inference is based. This inherent difficulty is a limitation of our approach, as well as of any other approach we are aware of.

In order to infer the potential interactions between genes, we ran the algorithm 200 times with population size set as 30 and examined the best solution of each run as we did for the artificial data. We evolved the network structures and the network weights for 300 and 2000 generations, respectively, and the parameter δ was set equal to 0.4, which limits the possible number of connectivity to 5-15. The initial values for the weights and bias were between -1 and 1, while the initial values for the decay rates were between 1 and 10.

Based on the curve depicted in Fig. 9, we observe that there are two positions suggesting the possible number of connections in the inferred network. If we choose p at 0.16, the estimated number of connections in the network is 16, which includes all the 9 true relations in the system, together with 7 false positives. If we operate in a more conservative way, we decide the possible

number of connections in the network is 7. In this case, 5 true positives are identified with only 2 false positives. The highest observance frequency is 52.5%, corresponding to the inhibition of *lexA* on *umuD*. The other 4 connections that are correctly identified include the inhibition of *lexA* on *uvrD*, *recA*, *uvrA*, and *polB*. The false positives are between the gene pairs *ruvA – lexA* and *uvrA – lexA*. Clearly, PSO/RNN can provide meaningful insight into unveiling the most significant connections in genetic regulatory networks. However, when we used more experimental time series, such as 2 or even all 4 series, we did not observe improved results. On the contrary, sometimes, the results degraded. The reason may be that the proposed model does not consider time delay and other potential factors, which are ubiquitous in gene regulatory activities. If, on the other hand, more time series are introduced into the model, which oscillate abruptly and are not smooth, the model needs to include more additional parameters in order to handle and interpret the increasing complexities. Experimental results for the SOS network using RNN trained with BPTT [19] and dynamic Bayesian networks [33] are also reported. They both correctly identify 4 gene relations with 3 positives. In this context, RNN/PSO unveils more true connections in the SOS networks, with fewer false positives, than the other two methods. However, due to the limitation of the available data, we expect the properties of the methods can be more effectively investigated with higher quality data.

V. CONCLUSIONS

One of the central tasks in molecular genetics is to understand gene regulatory mechanisms. Inference of genetic regulatory networks based on the time series gene expression data from microarray experiments has become an important and effective way to achieve this goal. Herein, we employ recurrent neural networks to model the regulatory systems, and further, to reveal the

potential interactions between genes. Particle swarm optimization is used for both network training and architecture identification. The simulation results on both the artificial and real data demonstrate that the proposed method is very promising in capturing the nonlinear dynamics of genetic regulatory systems and unveiling the potential gene interaction relation. Given the similarity between RNNs and gene networks, we believe that RNNs will play an important role in exploring the mystery of gene regulation relationships. However, currently, the major limiting factor for genetic regulatory network analysis is the paucity of reliable gene expression time series data. Current research can only focus on the modeling of networks from synthetic data, or the simulation of small-scale real networks, with only several genes or gene clusters. No attempt has been made to infer large-scale genetic regulatory networks due to this restriction. Having high quality time series gene expression data with a sufficient number of time points is particularly important in further investigating and evaluating the performance of current computational models. Naturally, the number of data points will be practically determined by the particular network and system being studied, and the smaller that interval, the better the resolution. However, the minimum time interval between data points to resolve individual effects between genes for a given genetic regulatory network will have to take into consideration the number of genes involved, the speed of the regulatory response, and the magnitude of the regulatory effects of one gene upon another. Protein and metabolite data also need to be combined with gene expression data to provide more reasonable and accurate modeling. The role that external variables play (affectors of expression that originate and act downstream from transcription) will vary from system to system. However, ignoring these variables at this stage will not invalidate the inferred regulatory network because the networks being studied, by definition, are largely controlled via transcription. When additional proteomic

and metabolomic data can be integrated into the data set, the inferred regulatory network will have the same basic relationships but with more details apparent.

For the RNN model, we consider several further extensions. First, genetic networks are known to be robust to noise, and gene expression levels in the regulatory systems will not be affected dramatically due to the small perturbation caused by some internal factors or the external environment. This characteristic raises the question of increasing the robustness and redundancy of current models. Second, time delay is an important property of genetic regulatory networks, which is, however, not well addressed yet. RNNs are powerful in dealing with temporal information and well suited to handle this type of problem. However, this will introduce more parameters into the model and asks for more training data. Generating a synthetic system that can simulate some well-known gene networks may allow for preliminary investigation. Last but not least, prior information about genes can be combined into the model in order to remove some impossible connections and simplify computation. For example, genes that are co-regulated may share similar expression patterns and have common motifs. This information can be used to eliminate false positives.

ACKNOWLEDGEMENT

Partial support for this research from the National Science Foundation, and from the M.K. Finley Missouri endowment, is gratefully acknowledged. The author would also like to thank the Editor and the anonymous reviewers for their valuable comments.

REFERENCE

[1] Z. Bar-Joseph, "Analyzing time series gene expression data," *Bioinformatics*, vol. 20, no. 16, pp. 2493-2503, 2004.

[2] Z. Bar-Joseph, G. Gerber, D. Gifford, T. Jaakkola, and I. Simon, "A new approach to analyzing gene expression time series data," In *Proceedings of the 6th Annual International Conference on Research in Computational Molecular Biology (RECOMB),* pp 39-48, 2002.

[3] X. Cai and D. Wunsch, "Engine data classification with simultaneous recurrent network using a hybrid PSO-EA algorithm," In *Proceedings of 2005 IEEE International Joint Conference on Neural Networks*, vol. 4, pp. 2319-2323, 2005.

[4] M. Dasika, A. Gupta, and C. Maranas, "A mixed integer linear programming framework for inferring time delay in gene regulatory networks," In *Proceedings of the Pacific Symposium on Biocomputing*, pp. 474-485, 2004.

[5] P. D'haeseleer, "Reconstructing gene network from large scale gene expression data," Dissertation, University of New Mexico, 2000.

[6] P. D'haeseleer, S. Liang, and R. Somogyi, "Genetic network inference: From co-expression clustering to reverse engineering," *Bioinformatics*, vol. 16, no. 8, pp. 707-726, 2000.

[7] P. D'haeseleer, X. Wen, S. Fuhrman, and R. Somogyi, "Linear modeling of mRNA expression levels during CNS development and injury," In *Proceedings of the Pacific Symposium on Biocomputing (PSB'99),* pp. 41-52, 1999.

[8] S. Doctor, G. Venayagamoorthy, and V. Gudise, "Optimal PSO for collective robotic search applications," In *Proceedings of Congress on Evolutionary Computation 2004*, vol. 2, pp. 1390-1395, 2004.

[9] R. Eberhart and Y. Shi, "Particle swarm optimization: Developments, applications and recourses," In *Proceedings of the 2001 Congress on Evolutionary Computation*, vol. 1, pp. 81-86, 2001.

[10] M. Eisen and P. Brown, "DNA arrays for analysis of gene expression," *Methods Enzymol*, vol. 303, pp. 179-205, 1999.

[11] D. Fogel, "An introduction to simulated evolutionary optimization," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 3-14, 1994.

[12] N. Friedman, M. Linial, I. Nachman, and D. Pe'er, "Using Bayesian networks to analyze expression data," *Journal of Computational Biology*, vol. 7, pp. 601-620, 2000.

[13] V. Gudise and G. Venayagamoorthy, "Comparison of particle swarm optimization and backpropagation as training algorithms for neural networks," In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pp. 110-117, 2003.

[14] J. Hallinan, "Cluster analysis of the p53 genetic regulatory network: Topology and biology. In *Proceedings of the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pp. 1-8, 2004.

[15] J. Hallinan and P. Jackway, "Network motifs, feedback loops and the dynamics of genetic regulatory networks," In *Proceedings of the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pp. 1-7, 2005.

[16] J. Hallinan and J. Wiles, "Evolving genetic regulatory networks using an artificial genome," In *Proceedings of the 2nd Asia-Pacific Bioinformatics Conference (APBC2004)*, vol. 29, pp. 291-296, 2004.

[17] J. Hallinan and J. Wiles, "Asynchronous dynamics of an artificial genetic regulatory network," In *Proceedings of the 9th International Conference on the Simulation and Synthesis of Living Systems (ALife9)*, 2004.

[18] S. Haykin, "Neural networks: A comprehensive foundation," 2nd Ed., Prentice Hall, New Jersey, 1999.

[19] X. Hu, A. Maglia, and D. Wunsch II, "A general recurrent neural network approach to model genetic regulatory networks," In *Proceedings of the 27th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pp. 4735-4738, 2005.

[20] D. Husmeier, "Sensitivity and specificity of inferring genetic regulatory interactions from microarray experiments with dynamic Bayesian networks," *Bioinformatics*, vol. 19, no. 17, pp. 2271- 2282, 2003.

[21] H. De Jong, "Modeling and simulation of genetic regulatory systems: A literature review," *Journal of Computational Biology*, vol. 9, pp. 67-103, 2002.

[22] C. Juang, "A hybrid of genetic algorithm and particle swarm optimization for recurrent network design," *IEEE Transaction on Systems, Man, and Cybernetics*, Part B, vol. 34, no. 2, pp. 997-1006, 2004.

[23] S. Kauffman, "The origins of order: Self-organization and selection in evolution," Oxford University Press, New York, 1993.

[24] E. Keedwell and A. Narayanan, "Discovering gene networks with a neural-genetic hybrid," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 2, no. 3, pp. 231-242, 2005.

[25] J. Kennedy and R. Eberhart, "A discrete binary version of the particle swarm optimization," In *Proceedings of International Conference on System, Man, and Cybernetics*, vol. 5, pp. 4104-4108, 1997.

[26] J. Kennedy, R. Eberhart, and Y. Shi, "Swarm intelligence," Morgan Kaufmann Publishers, 2001.

[27] J. Kolen and S. Kremer, "A field guide to dynamical recurrent networks," IEEE Press, 2001.

[28] S. Liang, S. Fuhrman, and R. Somogyi, "REVEAL: A general reverse engineering algorithm for inference of genetic network architectures," In *Proceedings of the Pacific Symposium on Biocomputing (PSB'98),* vol. 3, pp. 18-29, 1998.

[29] R. Lipshutz, S. Fodor, T. Gingeras, and D. Lockhart, "High density synthetic oligonucleotide arrays," *Nature Genetics*, vol. 21, pp. 20-24, 1999.

[30] G. McLachlan, K. Do, and C. Ambroise, "Analyzing microarray gene expression data," John Wiley & Sons, Inc., Hoboken, NJ, 2004.

[31] E. Mjolsness, T. Mann, R. Castaño, and B. Wold, "From co-expression to co-regulation: An approach to inferring transcriptional regulation among gene classes from large-scale expression data," In *Advances in Neural Information Processing Systems 12*, pp. 928-934, MIT Press, 2000.

[32] I. Ong, J. Glasner, and D. Page, "Modeling regulatory pathways in E.coli from time series expression profiles," In *Proceedings of the 10th International Conference on Intelligent Systems for Molecular Biology (ISMB02)*, pp. 1-8, 2002.

[33] B. Perrin, L. Ralaivola, A. Mazurie, S. Battani, J. Mallet, and F. d'Alchė-Buc, "Gene networks inference using dynamic Bayesian networks," *Bioinformatics*, vol. 19, Suppl.2, pp. ii138- ii148, 2003.

[34] M. Ronen, R. Rosenberg, B. Shraiman, and U. Alon, "Assigning numbers to the arrows: Parameterizing a gene regulation network by using accurate expression kinetics," *Proceedings of the National Academic Sciences*, vol. 99, no. 16, pp.10555-10560, 2002.

[35] M. Settles, B. Rodebaugh, and T. Soule, "Comparison of genetic algorithm and particle swarm optimizer when evolving a recurrent neural network," In *Proceedings of the Genetic and Evolutionary Computation Conference 2003*, vol. 2723, pp. 148-149, 2003.

[36] I. Shmulevich, E. Dougherty, and W. Zhang, "From Boolean to probabilistic Boolean networks as models of genetic regulatory networks," *Proceedings of the IEEE*, vol. 90, no. 11, pp. 1778- 1792, 2002.

[37] E. van Someren, L. Wessels, M. Reinders, "Linear modeling of genetic networks from experimental data," In *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB00)*, pp. 355-366, 2000.

[38] E. van Someren, L. Wessels, and M. Reinders, "Genetic network models: A comparative study," In *Proceedings of SPIE, Micro-arrays: Optical Technologies and Informatics (BIOS01)*, vol. 4266, pp. 236-247, 2001.

[39] E. van Someren, L. Wessels, M. Reinders, and E. Backer, "Robust genetic network modeling by adding noisy data," In *Proceedings of the 2001 IEEE-EURASIP Workshop on Nonlinear Signal and Image Processing (NSIP01),* 2001.

[40] Y. Tamada, S. Kim, H.Bannai, S. Imoto, K. Tashiro, S. Kuhara, and S. Miyano, "Estimating gene networks from gene expression data by combining Bayesian network model with promoter element detection," *Bioinformatics*, vol. 19, Suppl.2, pp. ii227- ii236, 2003.

[41] J. Vohradský, "Neural network model of gene expression," *The FASEB Journa*l, vol. 15, pp. 846-854, 2001.

[42] M. Wahde and J. Hertz, "Coarse–grained reverse engineering of genetic regulatory networks," *Biosystems*, 55, 129–136, 2000.

[43] M. Wahde and J. Hertz, "Modeling genetic regulatory dynamics in neural development," *Journal of Computational Biology*, 8, 429-442, 2001.

[44] M. Wahde and Z. Szallasi, "A survey of methods for classification of gene expression data using evolutionary algorithms," *Expert Review of Molecular Diagnostics*, vol. 6, no. 1, pp. 101-110, 2006.

[45] D. Weaver, C. Workman, and G. Stormo, "Modeling regulatory networks with weight matrices," In *Proceedings of the Pacific Symposium on Biocomputing*, pp. 112-123, 1999.

[46] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of IEEE*, vol. 78, no. 10, pp. 1550-1560, 1990.

[47] J. Xu and B. Nelson, Personal Communications, Department of Industrial Engineering and Management Sciences, Northwestern University, 2006.

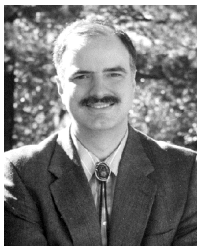
[48] R. Xu, X. Cai, and D. Wunsch II, "Gene expression data for DLBCL cancer survival prediction with a combination of machine learning technologies," In *Proceedings of the 27th Annual International Conference of IEEE Engineering in Medicine and Biology Society*, pp. 894-897, 2005.

[49] R. Xu, X. Hu, and D. Wunsch II, "Inference of genetic regulatory networks from time series gene expression data," In *Proceedings of International Joint Conference on Neural Networks 2004*, vol.2, pp.1215-1220, 2004.

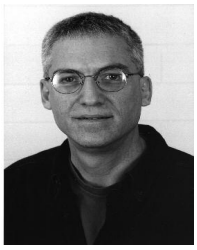
[50] R. Xu and D. Wunsch II, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol.16, no.3, pp.645-678, 2005.

[51] X. Yao, "Evolving artificial neural networks," *Proceedings of the IEEE*, vol. 87, no.9, pp.1423-1447, 1999.

Rui Xu received the B.E. degree in electrical engineering from Huazhong University of Science and Technology, Wuhan, Hubei, China, in 1997, the M.E. degree in electrical engineering from Sichuan University, Chengdu, Sichuan, in 2000, and the Ph.D. degree in electrical engineering from the University of Missouri - Rolla in 2006. His research interests include machine learning, neural networks, pattern classification and clustering, and bioinformatics. He is a member of the IEEE.



Donald C. Wunsch II received the B.S. degree in applied mathematics from the University of New Mexico, Albuquerque, and the M.S. degree in applied mathematics and the Ph.D. degree in electrical engineering from the University of Washington, Seattle.
He is the Mary K. Finley Missouri Distinguished Professor of Computer Engineering at the University of Missouri - Rolla, where he has been since 1999. His prior positions were Associate Professor and Director of the Applied Computational Intelligence Laboratory at Texas Tech University, Senior Principal Scientist at Boeing, Consultant for Rockwell International, and Technician for International Laser Systems. He has well over 200 publications and has attracted over $5 million in research funding. He has produced eleven Ph.D. recipients – six in electrical engineering, four in computer engineering, and one in computer science.
Dr. Wunsch has received the Halliburton Award for Excellence in Teaching and Research, and the National Science Foundation CAREER Award. He served as voting member of the IEEE Neural Networks Council, Technical Program Co-Chair for IJCNN 02, General Chair for IJCNN 03, International Neural Networks Society Board of Governors Member, and 2005 President of the International Neural Networks Society.



Ronald L. Frank received the B.S. degree in biology from Houghton College in Houghton, New York in 1978, the M.S. degree in 1981 and Ph.D. degree in genetics in 1985 from The Ohio State University, Columbus, OH. He is currently Associate Professor of Biological Sciences at University of Missouri-Rolla where he has been since 1988. Prior to UMR he was an ARS Post-doctoral Fellow at the Beltsville Agricultural Research Center in Maryland. Dr. Frank's research interests are in structure, expression and evolution of gene families in soybean. He teaches Molecular Genetics and Evolution classes at UMR and is a voting member of the Genetics Society of America.