Applying Intelligent Computing Techniques to Modeling Biological Networks from Expression Data

Wei-Po Lee[1]* and Kung-Cheng Yang[2]

[1] *Department of Information Management, National Sun Yat-sen University, Kaohsiung, Chinese Taipei;* [2] *Department of Management Information Systems, National Pingtung University of Science and Technology, Pingung, Chinese Taipei.*

Constructing biological networks is one of the most important issues in systems biology. However, constructing a network from data manually takes a considerable large amount of time, therefore an automated procedure is advocated. To automate the procedure of network construction, in this work we use two intelligent computing techniques, genetic programming and neural computation, to infer two kinds of network models that use continuous variables. To verify the presented approaches, experiments have been conducted and the preliminary results show that both approaches can be used to infer networks successfully.

Key words: reverse engineering, system modeling, genetic programming, recurrent neural network, expression data

Introduction

Systems biology studies the interactions between the components of a biological system. The study investigates how these interactions give rise to the function and behavior of that system. It aims to understand biological systems at a system-level, and focuses mainly on unraveling molecular systems at the level of pathways and the group of pathways in a cell and its neighboring cells (*1*, *2*). In systems biology research, system structure and system dynamics are the two core properties of a biological system to be studied. System structure means the studies of networks of gene/protein interactions, biochemical pathways, and the mechanisms to modulate the physical properties of intra-cellular and inter-cellular structures; and system dynamics concerns the dynamical behavior of a system over time under various conditions. To address the above two issues, biologists and computational scientists have been working on creating and exploring predictive dynamical models of complex biological systems such as metabolic, gene-regulation, or signal-transduction pathways in living cells. With the network models, we can now uncover some complex behavior patterns by constructing networks from measured time series data, and then analyzing and studying the interactions between interconnected components in a network.

***Corresponding author.**
E-mail: wplee@mail.nsysu.edu.tw

However, constructing a network from data manually takes a considerable large amount of time, therefore an automated procedure is advocated. Reverse engineering is a paradigm with great promise for analyzing and constructing biological networks (*2–4*). The procedure involves altering the network in some ways, observing the outcome, and using mathematics and logic to infer the underlying principles of the network. It is concluded that the key issues of this process lie in selecting network model and fitting network parameters and structures into the available data.

Many molecular-interaction models have been proposed and implemented by using various formalisms (*1*, *5*). They include differential equations or stochastic molecular simulation formalism, and range from simple models of mass action and Michaelis-Menten kinetics to more complex models of enzyme reactions and gene regulation. Michaelis-Menten kinetic model is proposed to describe enzymatic catelysis (*6*). In this model, a kinetic equation is defined as a function of substrate and/or product concentration. This model has become a cornerstone of much of the modern analysis of enzyme reaction mechanism. On the other hand, gene regulation is a complex process in the synthesis of proteins that may function as transcription factors binding to regulatory sites of other genes, as enzymes catalyzing metabolic reactions, or as components of signal transduction pathways (*5*, *7*, *8*).

According to the types of variables used in the modeling procedure, biological network models can mainly be categorized into two types, discrete and continuous ones (*5*, *9*, *10*). The first type of models uses discrete variables and assumes that genes only exist in discrete states. This approximation is usually implemented by Boolean variables in which the gene is in either on or off state. Boolean networks are easy to simulate in a cheaper computational cost, but it has been proven that Boolean networks are not able to capture some system behaviors that can only be observed on continuous models. Another popular discrete variable model is Bayesian network that explicitly establishes probabilistic relationships between nodes (*11*, *12*). Bayesian models have rich statistics and probability semantics, but learning network structure for such models is computationally expensive. In addition, Bayesian models are inherently static. As the directed network graphs are acyclic by definition, there can be no auto-regulation and no time-course regulation.

In addition to using variables with discrete states, the second type of models uses continuous variables. One of the popular continuous variable models is the one based on differential equations that can describe more accurately the system dynamics of a gene regulatory network (GRN) (*13*, *14*). Compared with discrete variable models, the differential equation models can more accurately represent the underlying physical phenomena due to its continuous variables. In addition, there are many theories of system analysis and of control design on dynamical systems to support this type of models. However, it should be noted that the non-linear ordinary differential equations are hard to solve. It is too difficult for the traditional optimization methods to estimate all the large number of parameters involved in a GRN. The other commonly used continuous variable model is the neural network-based model, among which the recurrent neural networks (RNNs) are the most successful ones (*15*, *16*). This model is biologically plausible and noise-resistant. It is continuous in time, and uses a transfer function to transfer the inputs into a shape close to the one observed in natural processes. Also its non-linear characteristics provide information about the principles of control and natural interactions of elements of the modeled system.

As is analyzed above, different models have been proposed to simulate biological networks, and computational methods have also been developed to reconstruct networks from the temporal measured data correspondingly. More details can be found in the literature (*5*, *9*, *10*), where it can be seen that the works in modeling networks shared similar ideas in principle. However, depending on the research motivations behind the work, different researchers explored the same topic from different points of view; thus the implementation details of individual work are different. Instead of subjectively arguing which approach is better to offer for network reconstruction, our work here focuses on investigating whether the presented approach, in practice, can be used to model biological networks, and on how to develop complex networks. Because continuous variables can more accurately describe the underlying physical phenomena of the biological systems to be modeled, in this work we only consider models with continuous variables. We take the two most representative models, kinetic equations and neural networks, to represent biological networks, and employ two intelligent computing techniques, genetic programming and neural computation, to reconstruct systems from collected data respectively. In order to deal with the scalability problem in modeling gene regulatory networks, a clustering method with some data analysis techniques for feature extraction is developed to construct networks in a hierarchical way. To verify the presented approaches, different experiments have been conducted to demonstrate how they work. The results show the promise of our approaches.

Models and Methods

Constructing networks from measured data involves two major steps: choosing a network model that is biologically and computationally plausible, and adopting an appropriate computing method to build the network model reversely from the measured time series data. This section describes how we take two models that deal with continuous variables to represent different biological networks and then employ two intelligent computing methods to automatically create the corresponding networks respectively.

Constructing networks by evolutionary computation

Kinetic equation-based network model

Modern biology researchers have regarded biological systems as networks. From the simple pair-wise to complex regulatory interactions, all can be rep-

resented as networks. As indicated, the core of the network study lies in the interactions of the entities (components) of a biological system and in investigating how these interactions give rise to the function and behavior of that system. All networks share some common characteristics, and mathematical methods have been established to describe network structure and interactions of network components. The interactions between network components can be represented by some basic chemical reaction schemes, in which the concentrations of substrates, catalysts (such as enzymes), intermediate substrates, and products participating in chemical reactions are often modeled by non-linear continuous-time differential equations. It has been shown that by identifying the most highly connected components, the network topology can be inferred from expression data. For networks with enzyme-catalyzed reactions, we take the form of Michaelis-Menten kinetics shown below as the network model (*6*), since it is useful in understanding the mechanism by which an enzyme carries out its catalytic activity. The components involved in the reactions can be represented as (*17*, *18*):

$$\mathrm{E} + \mathrm{S} \underset{k_{-1}}{\overset{k_1}{\rightleftarrows}} \mathrm{ES} \overset{k_2}{\rightarrow} \mathrm{E} + \mathrm{P}$$

$$\begin{aligned}
\tfrac{d[E]}{dt} &= -k_1[E][S] + (k_{-1} + k_2)[ES] \\
\tfrac{d[S]}{dt} &= -k_1[E][S] + k_{-1}[ES] \\
\tfrac{d[ES]}{dt} &= k_1[E][S] - (k_{-1} + k_2)[ES] \\
\tfrac{d[P]}{dt} &= k_2[ES]
\end{aligned}$$

The above equations are specified by the rate constants k_i and the initial concentrations of enzyme E, substrate S, and product P. The terms k_1, k_{-1}, and k_2 are rate constants for the association of substrate and enzyme, the dissociation of unaltered substrate from the enzyme, and the dissociation of product from the enzyme, respectively. During the time course analysis, addition of other reactants or alterations of kinetic parameters can be accommodated within this set of equations. With such a network model, computational methods can be used to derive the relevant equations. In this work we use an evolutionary approach, namely genetic programming (GP) to automatically create the equations (that is, the right hand sides of the above equations). The reaction rate of each network component is determined by other relevant components genetically. Different from the use of string representation in genetic algorithms, GP-based approaches take a tree-like structure as their representation. Consequently, using GP to evolve kinetic equations has the advantage of operating variable-size genotypes. This is an important feature because it provides complete freedom for the kinetic equations in respect to the complexity of reactions that is generally difficult to predict in other methods.

Genetic programming

GP is an evolutionary computation technique invented by Koza (*19*), and its popularity is now increasing in the community of evolutionary computation research. It is an extension of the traditional genetic algorithms with the basic distinction that in GP, the individuals are dynamic tree structures rather than fixed-length vectors. GP aims to evolve dynamic and executable structures often interpreted as computer programs to solve problems without explicit programming. As in computer programming, a tree structure in GP is constituted by a set of non-terminals as the internal nodes of the trees, and by a set of terminals as the external nodes (leaves) of the trees. The construction of a tree is based on the syntactical rules that extend a root node to appropriate symbols (non-terminals/terminals) and each non-terminal is extended again by suitable rules accordingly, until all the branches in a tree end up with terminals. Hence, the first step in employing GP to solve a problem is to define appropriate non-terminals, terminals, and the syntactical rules associated for the program development. The search space in GP is the space of all possible tree structures composed of non-terminals and terminals.

According to our design, a tree structure of a kinetic equation has three parts: a dummy root node, the internal nodes, and the external nodes. The dummy root node is a non-terminal node. It is defined to collect the computing result of a tree equation, and works only for convenient manipulation by a GP system. The internal nodes are non-terminals as well; they are the common arithmetic operators, such as "+", "−", "×", "/". These nodes are defined to combine network components to form a tree equation. Finally, the external nodes (terminals) are network components that include all possible substances. They are defined to be the main ingredients of an equation. Also a terminal symbol R is defined to represent the set of possible numerical constants. Whenever a terminal symbol R appears in the tree creation procedure, a random number is generated to associate with R accordingly. Figure 1 shows some typical examples

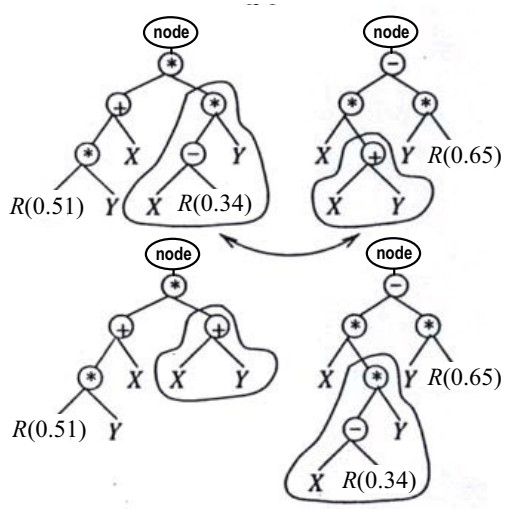


Fig. 1 The crossover operation in GP approach.

of an equation tree. In this figure, the upper two trees represent possible equations of reaction rate for a specific substance, which are interpreted as $[(0.51 \times Y)+X]\times[(X-0.34)\times Y]$ and $X\times(X+Y)-(Y\times 0.65)$, respectively.

The next step is to evaluate tree-individuals to determine their fitness for the creation of a new population. This is normally done by pre-defining a fitness function that quantitatively describes the requirements of a target task first, and then by executing the corresponding codes for tree-individuals in the environment of the particular problem. Here, the fitness function is to measure how the constructed model is close to the original system by calculating and comparing the outputs of the two systems. After that, the genetic operators are applied to the selected fitters (based on a certain selection criterion) to generate new trees. The evaluation and recreation cycle is repeated until the termination criterion is met.

In GP, three kinds of genetic operators—reproduction, crossover, and mutation—are normally used to create new tree individuals. Reproduction simply copies the original parent tree to the next generation; crossover randomly swaps sub-trees for two parents to generate two new trees; and mutation randomly regenerates a sub-tree for the original parent to create a new individual. Among them, crossover is the major one to create most of the offspring; when it is performed, all syntactic constraints must be satisfied to guarantee the correctness of new trees. Figure 1 shows an example of the operation of crossover in GP. Considerable details about GP are referred to Koza's work (*19*).

Constructing networks by neural computation

Structure-based network model

The behavior expressions of a GRN are in fact coordinated patterns of activities in time and space. Hence, GRNs can be regarded as dynamical systems that are perturbed by their interaction with the environment. RNNs are appropriate choices working as GRNs, because they have been shown to operate as dynamical systems that do not explicitly perform input-output mappings as other computational mechanisms. RNNs have recurrent connections that provide the possibility of generating oscillatory and periodic activities. The complex activities can be coordinated in the time domain, and the network behaviors can be governed by a set of differential equations.

There are several RNN architectures, ranging from restricted classes of feedback to full interconnection between nodes. Vohradsky and colleagues have proposed the use of fully RNN architecture in studying GRNs such as those involved in the transcriptional and translational control of gene expression (*15*, *16*). In this work, we also take the same architecture to model GRNs, but unlike their work that mainly simulates regulatory effects, our goal is to establish an approach to reconstruct regulatory networks from expression data measured.

In a fully recurrent net, each node has a link to any node of the net including itself. Using such a model to represent a gene regulatory network is based on the assumption that the regulatory effect on the expression of a particular gene can be expressed as a neural network in which each node represents a particular gene and the wiring between the nodes defines regulatory interactions. In a gene regulatory network, the level of expression of genes at time t can be measured from a gene node, and the output of a node at time $t + \Delta t$ can be derived from the expression levels and connection weights of all genes connected to the given gene at the time t. That is, the regulatory effect to a certain gene can be regarded as a weighted sum of all other genes that regulate this gene. Then the regulatory effect is transformed by a sigmoidal transfer function into a value between 0 and 1 for normalization.

The same set of the above transformation rules is applied to the system output in a cyclic fashion until the input does not change any further. As in Vohradsky's work (*16*), here we use the basic ingredient to

increase the power of empirical correlations in signaling constitutive regulatory circuits. It is to generate a network with nodes and edges corresponding to the level of gene expression measured in microarray experiments, and to derive correlation coefficients between genes. To calculate the expression rate of a gene, the following transformation rules are used:

$$\frac{dy_i}{dt} = k_{1,i}\ G_i - k_{2,i}\ y_i$$

$$G_i = \{1 + e^{-\left(\sum_j w_{i,j}\ y_j + b_i\right)}\}^{-1}$$

where y_i is the actual concentration of the i^{th} gene product; $k_{1,i}$ and $k_{2,i}$ are the accumulation and degradation rate constants of gene product, respectively; G_i is the regulatory effect on each gene that is defined by a set of weights $w_{i,j}$ estimating the regulatory influence of gene j on gene i, and an external input b_i representing the reaction delay parameter.

Learning algorithm

After the network model is decided, the next phase is to find settings of the thresholds and time constants for each neuron as well as the weights of the connections between the neurons so that the network can produce the most approximate system behavior (that is, the measured expression data). By introducing a scoring function for network performance evaluation, the above task can be regarded as a parameter estimation problem with the goal of maximizing the network performance (or minimizing an equivalent error measure). To achieve this goal, here we use the backpropagation through time (BPTT) (*20*) learning algorithm to update the relevant parameters of recurrent networks in discrete-time steps.

Instead of mapping a static input to a static output as in a feedforward network, BPTT maps a series of inputs to a series of outputs. The central idea is the "unfolding" of the discrete-time recurrent neural network (DTRNN) into a multilayer feedforward neural network when a sequence is processed. Figure 2 shows a typical example of unfolding an RNN. Once a DTRNN has been transformed into an equivalent feedforward network, the resulting feedforward network can then be trained using the standard backpropagation algorithm.

The goal of BPTT is to compute the gradient over the trajectory and update network weights accordingly. As mentioned above, the gradient decomposes over time. It can be obtained by calculating the instantaneous gradients and accumulating the effect over time. In BPTT, weights can only be updated after a complete forward step during which the activation is sent through the network and each processing element stores its activation locally for the entire length of the trajectory. More details on BPTT are referred to Werbos' work (*20*).

In the above learning procedure, learning rate is an important parameter. Yet it is difficult to choose an appropriate value to achieve an efficient training, because the cost surface for multi-layer networks can be complicated and what works in one location of the cost surface may not work well in another location. Delta-bar-delta is a heuristic algorithm for modifying the learning rate in the training procedure (*21*). It is inspired by the observation that the error surface may have a different gradient along each weight direction so that each weight should have its own learning rate. In our modeling work, to save the effort in choosing appropriate learning rate, this algorithm is implemented for automatic parameter adjustment.

Gene clustering

As can be expected, when the number of genes in a regulatory network and the interactions between the

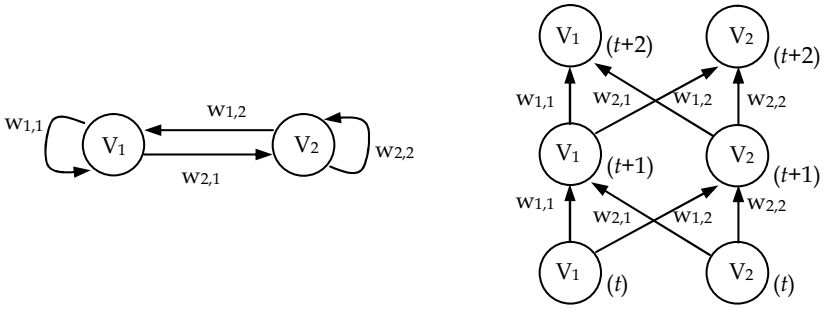


Fig. 2 The unfolding of a two-node recurrent neural network.

genes increase in respect to the increasing functional complexity that the network has to deal with, the direct modeling for a network becomes more and more difficult to achieve. To scale up our approach to solve more complicated reconstruction task, we take an engineering point of view to tackle the problem in a "divide and conquer" way. A clustering technique is firstly employed to group the genes into some small-scale networks, based on the analysis of their corresponding expression data, and then the small networks are reconstructed from the expression data by the RNN-based approach described above. Once all the small networks have been obtained, each of them is regarded as a self-contained system component of the original network, and the learning algorithm indicated previously is used to determine the interactions between different system components. The small networks can be decomposed again in the similar way until the resulting networks can be directly modeled.

In our current work, the self-organization feature map (SOM) method is adopted for gene clustering. Before a clustering method is applied to the expression data, some features on the dataset have to be decided so that the clustering method can find the relationships between the data accordingly. Here we use the wavelet transform (WT) technique to extract data features from the waveforms derived from the gene expression data of different time points.

The WT theory has been widely used in many signal-processing applications (*22*). WT decomposes a signal into a set of basis functions called wavelets. It involves representing a time function in terms of simple and fixed building blocks. These building blocks are actually a family of functions derived from a single generating function (the mother wavelet) by translation and dilation operations. It is known that WT is more suitable in analyzing non-stationary signals, since it is well localized in time and frequency (*22*). With its important ability on data manipulation, WT can compress an original signal that consists of many data points into a few parameters called wavelet coefficients that characterize the behavior of the signal. The wavelet coefficients can be computed by using the discrete wavelet transform. The computed wavelet coefficients provide a compact representation that shows the energy distribution of the signal in time and frequency. Therefore, the wavelet coefficients derived from the time-varying gene regulatory signals can be used as features of the signals for gene clustering.

Figure 3 is the typical result of wavelet transform for a certain gene (produced by MATLAB Wavelet toolbox), in which S is the original gene expression data; a_4 is the wavelet approximation (taken from the Daubechies function with wavelets of order 4) by the relevant subsequences; and d_1 to d_4 are the wavelet detailed subsequences (with four levels of multi-resolution analysis). The coefficients of the high frequent wavelet subsequences are then used as data features for SOM clustering.

Evaluation

After presenting the two models and proposing two intelligent computing methods for building biological networks respectively, we conduct two sets of experiments to evaluate our methodology. The first is to investigate whether the GP approach can evolve the

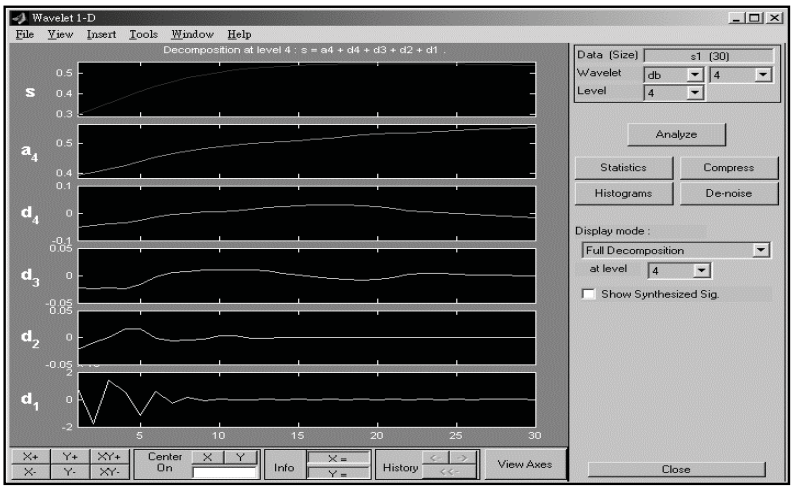


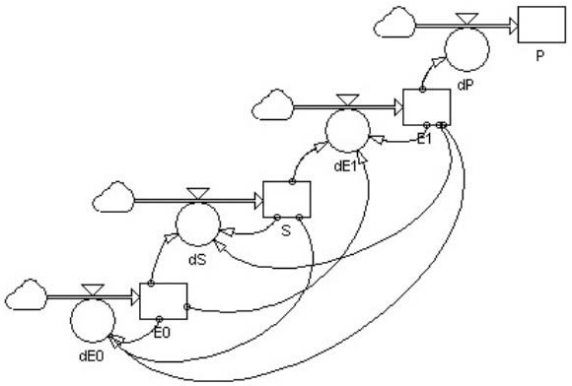
Fig. 3 The wavelet transform for the expression data of a gene.

equations for a network, and the second is to reconstruct a network structure from data by neural computation approach.

Modeling equation-based network

The first experiment is to examine the performance of using GP to evolve kinetic equations. To obtain time series data, we first used the well-known simulation software POWERSIM to create a system dynamics model for the kinetic equations listed above. Then we used the model developed in the simulated environment to produce expression data for five steps and the data points were then interpolated up to fifty. In the simulation, the rate constants k_1, k_{-1}, and k_2 were set to 10.0, 1.0, and 5.0; and the initial concentrations for enzymes E0, immediate substance E1, substrate S, and product P were 0.08, 0, 1.0, and 0, respectively. Figure 4 shows the system dynamics model and the reaction equation of the original network and its corresponding behavior during the simulation period.

To evolve the structure of kinetic equation for each substance, in the experiment we defined the non-terminal set as {node, +, −, ×, /} that includes the dummy root node and the common arithmetic operators, and the terminal set as {E0, E1, S, P, R} that includes all possible substances. In the terminal set, R is the set of real numbers between −10 and 10 that represents the possible numerical constants. As described above, each R in the tree is attached with a random numerical value within the specified range to represent a constant. The fitness function here was in fact a penalty function that accumulated the error (the difference between desired and actual time series data at each time point) produced by each individual (a kinetic equation) over 50 simulated time steps. In each experimental run, one population of 500 individuals was used and the evolution process continued for 50 generations. Figure 5 shows the typical behavior of the evolutionary mechanism. Figure 6 is two examples illustrating how the error varies in evolving equations for substances E0 and E1. As can be seen, the error is reduced gradually by the method used, and the equations obtained are almost identical to the original ones. They indicate that the equation-based network model can be built successfully by the GP approach.



$$\mathrm{E0} + \mathrm{S} \underset{k_{-1}}{\overset{k_1}{\rightleftarrows}} \mathrm{E1} \xrightarrow{k_2} \mathrm{E0} + \mathrm{P}$$

Fig. 4 Illustration of the original network.

Modeling RNN-based network

To evaluate our approach for the network-based model, we firstly used the well-known gene regulatory network simulation software Genexp (*16*) to produce expression data, and then employed our computational approach to infer network models from the data. A four-node network was defined in which the accumulation and degradation rate constants of gene product k_1 and k_2 for all genes were all set to 0.3 (chosen from preliminary test). Figure 7 compares the system behaviors of the original and reconstructed networks. As can be seen, after training, the RNN can

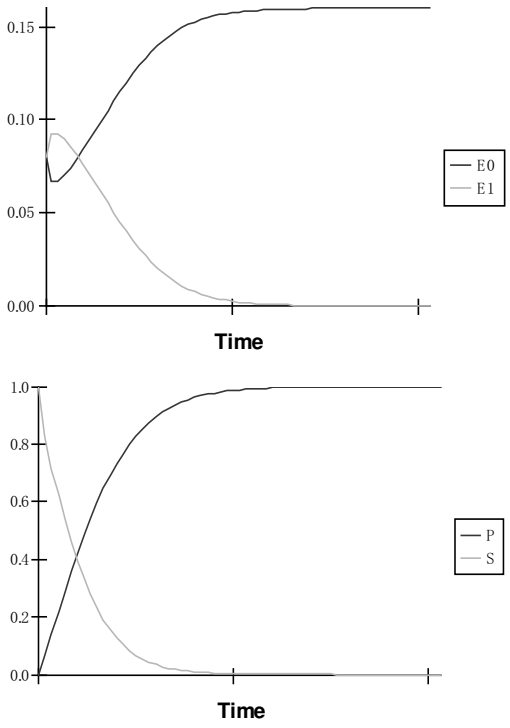


Fig. 5 The behavior of the original network. The X-axis and Y-axis are time step and product concentration, respectively.

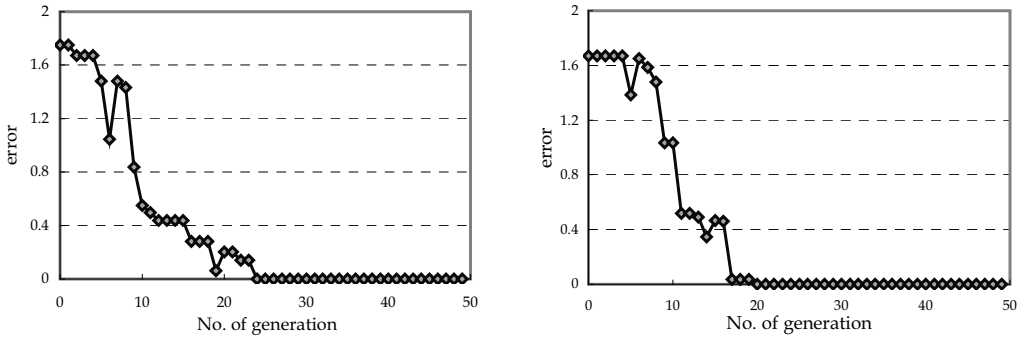


Fig. 6 Two examples of error curves (for E0 and E1) during the evolutionary process.

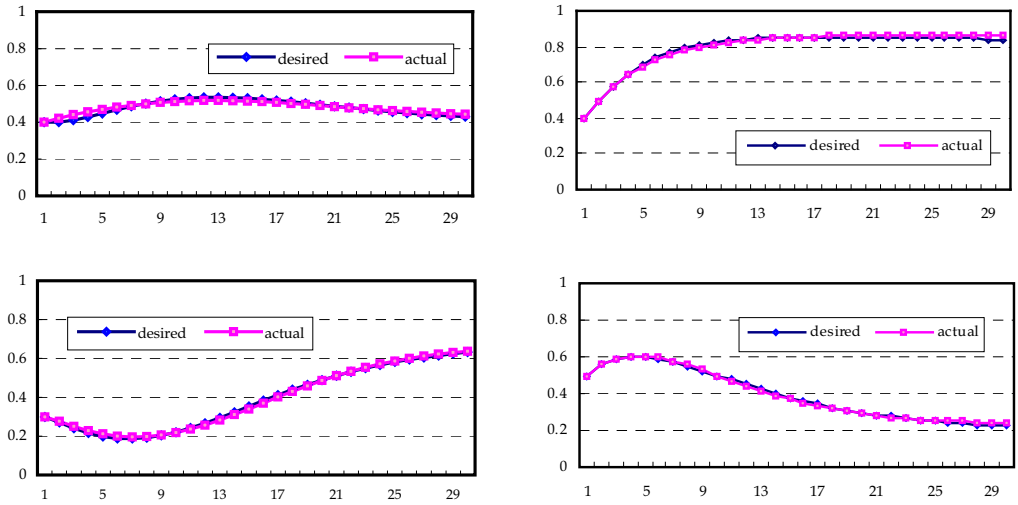


Fig. 7 Behaviors of the target and synthesized systems for the first dataset. The x-axis and Y-axis are time step and product concentration, respectively.

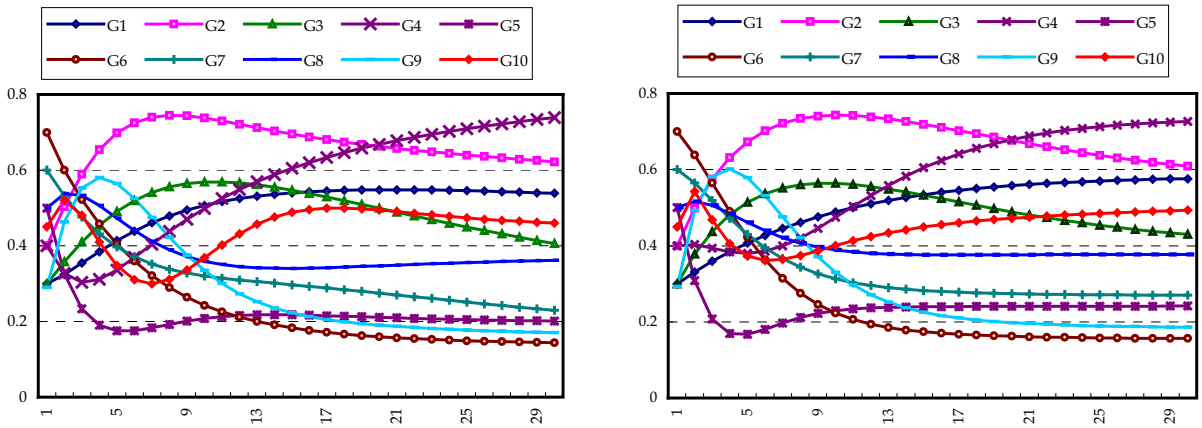


Fig. 8 Behaviors of the target (A) and rebuilt (B) systems for the second dataset. The X-axis and Y-axis are time step and product concentration, respectively.

successfully learn the system behavior of the original four-node regulatory network.

In modeling large systems with more genes, the data available are usually not sufficient to determine accurately the interactions between all genes in a given dataset. Hence, it is important to be able to construct a coarse-grained description of the system at first. The second experiment demonstrates how the clustering method can help inferring coarse-grained network models from data. The dataset was an ar-

tificial one obtained from the software Genexp. To collect data, initial parameters were specified for a 10-gene network and then the expression data were recorded for 30 consecutive time points (Figure 8A). To reconstruct the original network from these time series data, the gene clustering method including procedures of wavelet transform and SOM was used to group genes. Two clusters were observed, one for genes 1, 2, and 3, the other for genes 5, 6, 7, 8, and 9. Genes 4 and 10 did not obviously belong to any of the above two clusters. After furthermore measuring the gene distance and calculating the Pearson's correlation coefficients between the genes, we decided to organize the genes into four parts: two clusters as described above and genes 4 and 10 were considered as separated outliers.

Once the genes were grouped, the two clusters were both represented as two fully RNNs and trained independently. With the trained results, the overall four-part network was trained again together at a higher level. Figure 8B presents the system behaviors of the original and reconstructed networks over the 10 genes. Though this is not a perfect match in data fitting, it can be observed that the behavior of the trained network is very similar to the original one, in which many of them have almost identical data sequences. This is satisfactory as the dataset is relatively small in fact.

Conclusion

The construction of biological networks is one of the most important issues in systems biology. Many models have been proposed to simulate networks, and computational methods have also been developed to reconstruct networks. Instead of arguing which model and method are most suitable for network reconstruction, in this work we emphasize the importance of establishing a practical approach that can model biological networks and is scalable for inferring complex networks. As kinetic equations can describe biochemical reactions in continuous time, we therefore choose them to model networks of this kind, and employ the GP method to evolve equations. In addition, recurrent networks can work as dynamical systems as GRNs do, so we adopt the RNN model to represent gene regulatory networks and use the neural computation method to learn networks. In order to deal with the scalability problem, a clustering method with several data analysis techniques for feature extraction has been developed to infer large networks hierarchically. To verify the presented approach, experiments have been conducted to demonstrate how it works for inferring small and large networks. The results have shown that our approach can be successfully used to infer networks from measured expression data.

Our work presented here directs to some prospects of future research. The first is to incorporate biological knowledge into our approach to construct networks (especially GRNs) in an even more efficient way. Biological knowledge about the general properties of genetic networks can alleviate some of the data requirements. If we take it into account in network reconstruction, it can thus reduce computational effort and obtain more accurate model. The other direction is to conduct more experiments with real biological datasets to furthermore evaluate our approaches. In addition, it is worth to investigate how to adopt other types of learning algorithms to improve the modeling performance. We are currently implementing a hybrid framework to evolve neural networks and to evaluate its corresponding performance.

Acknowledgements

This work was supported by National Science Council under contract NSC-93-2213-E-390-002.

Authors' contributions

WPL supervised the project, developed the computational algorithms, and wrote up the manuscript. KCY collected the datasets, conducted data analyses, and performed the experimental computation. Both authors read and approved the final manuscript.

Competing interests

The authors have declared that no competing interests exist.

References

1. Cheng, J., *et al.* 2005. Sigmoid: a software infrastructure for pathway bioinformatics and systems biology. *IEEE Intell. Syst.* 20: 68-75.
2. Kitano, H. 2001. Systems biology: toward system-level understanding of biological systems. In *Foundations of Systems Biology* (ed. Kitano, H.), pp.1-36. MIT Press, Cambridge, USA.

3. Csete, M.E. and Doyle, J.C. 2002. Reverse engineering of biological complexity. *Science* 295: 1664-1669.
4. Liao, J.C., *et al.* 2003. Network component analysis: reconstruction of regulatory signals in biological systems. *Proc. Natl. Acad. Sci. USA* 100: 15522-15527.
5. de Jong, H. 2002. Modeling and simulation of genetic regulatory systems: a literature review. *J. Comput. Biol.* 9: 67-103.
6. Michaelis, M.L. and Menten, L. 1913. The kinetics of invertin action. *Biochemische Zeitschrift* 49: 333-369.
7. Lewin, B. 1999. *Genes VII.* Oxford University Press, Oxford, UK.
8. Bower, J.M. and Bolouri, H. 2001. *Computational Modeling of Genetic and Biochemical Networks.* MIT Press, Cambridge, USA.
9. Styczynski, M.P. and Stephanopoulos, G. 2005. Overview of computational methods for the inference of gene regulatory networks. *Comput. Chem. Eng.* 29: 519-534.
10. D'haeseleer, P., *et al.* 2000. Genetic network inference: from co-expression clustering to reverse engineering. *Bioinformatics* 16: 707-726.
11. Hartemink, A.J. *et al.* 2002. Bayesian methods for elucidating genetic regulatory networks. *IEEE Intell. Syst.* 17: 37-43.
12. Ong, I.M., *et al.* 2002. Modeling regulatory pathways in *E. coli* from time series expression profiles. *Bioinformatics* 18: S241-248.
13. Kikuchi, S., *et al.* 2003. Dynamic modeling of genetic networks using genetic algorithm and S-system. *Bioinformatics* 19: 643-650.
14. Kimura, S., *et al.* 2005. Inference of S-system models of genetic networks using a cooperative coevolutionary algorithm. *Bioinformatics* 21: 1154-1163.
15. Blasi, M.F., *et al.* 2005. A recursive network approach can identify constitutive regulatory circuits in gene expression data. *Physica A* 348: 349-370.
16. Vohradsky, J. 2001. Neural network model of gene expression. *FASEB J.* 15: 846-854.
17. Bhalla, U.S. and Lyengar, R. 1999. Emergent properties of networks of biological signaling pathways. *Science* 283: 381-397.
18. Voit, E.O. 2000. *Computational Analysis of Biochemical Systems.* Cambridge University Press, Cambridge, UK.
19. Koza, J.R. 1992. *Genetic Programming*: *On the Programming of Computers by Means of Natural Selection.* MIT Press, Cambridge, USA.
20. Werbos, P.J. 1990. Backpropagation through time: what it does and how to do it. *Proc. IEEE* 78: 1550-1560.
21. Jacobs, R.A. 1988. Increased rates of convergence through learning rate adaptation. *Neural Networks* 1: 295-307.
22. Daubechies, I. 1990. The wavelet transform, time-frequency localization and signal analysis. *IEEE Trans. Inf. Theory* 36: 961-1005.