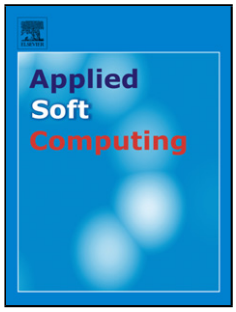


Title: Inferring Causal Networks using Fuzzy Cognitive Maps and Evolutionary Algorithms with Application to Gene Regulatory Network Reconstruction

Author: Ye Chen Lawrence J. Mazlack Ali A. Minai Long J. Lu

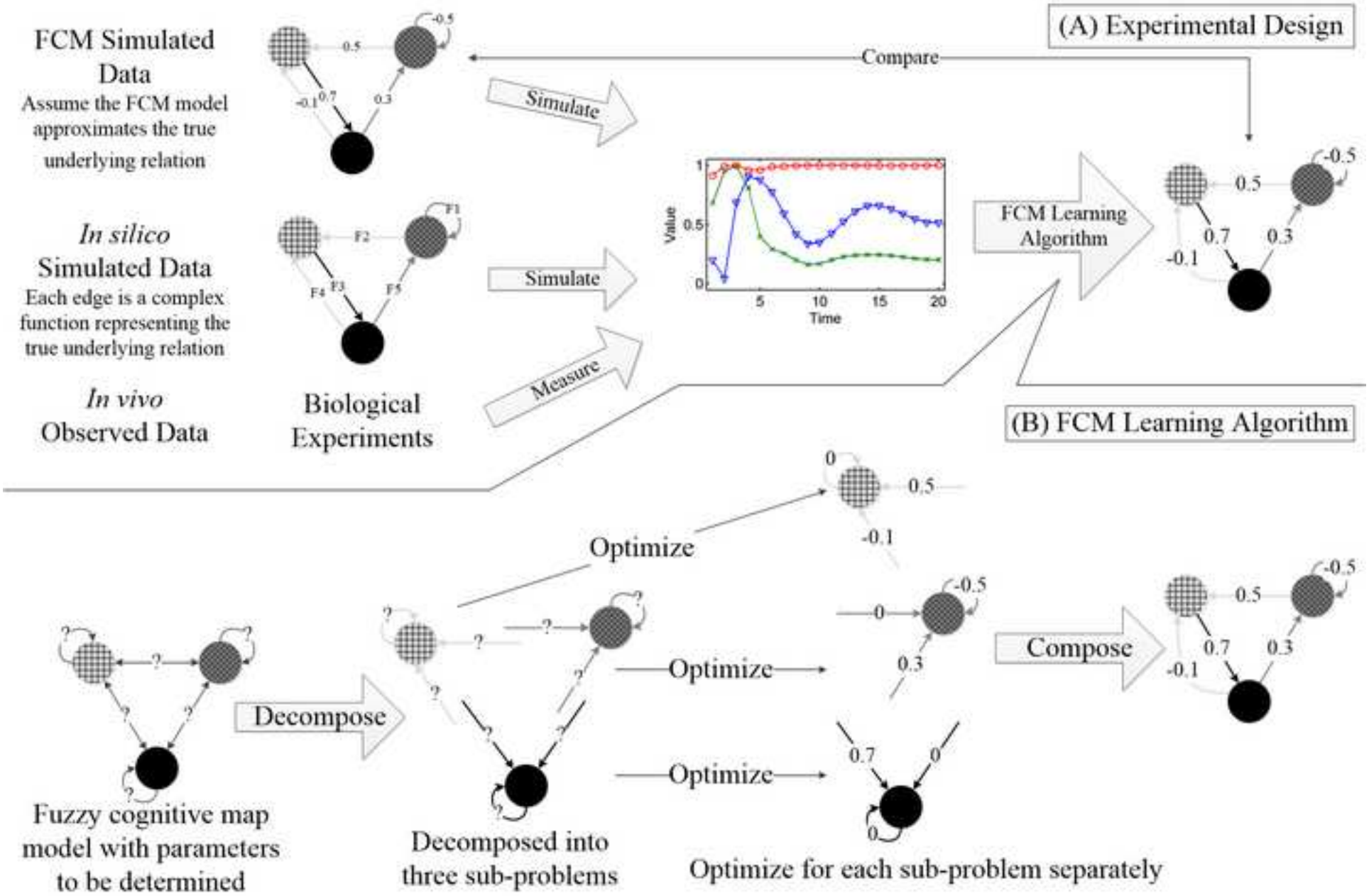
PII: S1568-4946(15)00540-2
DOI: http://dx.doi.org/doi:10.1016/j.asoc.2015.08.039
Reference: ASOC 3163

To appear in: *Applied Soft Computing*

Received date: 4-9-2014
Revised date: 2-8-2015
Accepted date: 14-8-2015

Please cite this article as: Y. Chen, L.J. Mazlack, A.A. Minai, L.J. Lu, Inferring Causal Networks using Fuzzy Cognitive Maps and Evolutionary Algorithms with Application to Gene Regulatory Network Reconstruction, *Applied Soft Computing Journal* (2015), http://dx.doi.org/10.1016/j.asoc.2015.08.039

*Graphical abstract (for review)



Highlights

- Proposed a decomposed evolutionary algorithm for learning fuzzy cognitive map models;
- Applied the proposed method to infer causal networks from gene expression time series;
- The algorithm is demonstrated to learn fuzzy cognitive maps with 300 nodes, while the accuracy of most existing methods is not satisfactory even for fuzzy cognitive maps with 40 nodes;
- Compared four different stochastic optimization algorithms for learning fuzzy cognitive maps, including genetic algorithm, ant colony optimization, differential evolution and particle swarm optimization.

Inferring Causal Networks using Fuzzy Cognitive Maps and Evolutionary Algorithms with Application to Gene Regulatory Network Reconstruction

Ye Chen [1, 2], Lawrence J. Mazlack [2], Ali A. Minai [2], and Long J. Lu[1, 3, 4, *]

08/01/2015

1. Division of Biomedical Informatics, Cincinnati Children's Hospital Research Foundation, 3333 Burnet Avenue, Cincinnati, OH 45229-3026

2. School of Electronics and Computing Systems, University of Cincinnati, 497 Rhodes Hall, Cincinnati, OH 45221

3. School of Computing Sciences and Informatics, University of Cincinnati, 810 Old Chemistry, Cincinnati, OH 45221-0008

4. Department of Environmental Health, College of Medicine, University of Cincinnati, 231 Albert Sabin Way, Cincinnati, OH 45267-0524

* Corresponding author:

Long J. Lu, Ph.D., Assistant Professor
Division of Biomedical Informatics, MLC 7024
Cincinnati Children's Hospital Research Foundation
3333 Burnet Avenue
Cincinnati, OH 45229

Phone: (513) 636-8720
Fax: (513) 636-2056
Email: long.lu@cchmc.org
Website: http://dragon.cchmc.org

Key words: Evolutionary algorithms, fuzzy cognitive map, gene regulatory networks, learning algorithm, optimization

ABSTRACT

Fuzzy cognitive maps have been widely used as abstract models for complex networks. Traditional ways to construct fuzzy cognitive maps rely on domain knowledge. In this paper, we propose to use fuzzy cognitive map learning algorithms to discover domain knowledge in the form of causal networks from data. More specifically, we propose to infer gene regulatory networks from gene expression data. Furthermore, a new efficient fuzzy cognitive map learning algorithm based on a decomposed genetic algorithm is developed to learn large scale networks. In the proposed algorithm, the simulation error is used as the objective function, while the model error is expected to be minimized. Experiments are performed to explore the feasibility of this approach. The high accuracy of the generated models and the approximate correlation between simulation errors and model errors suggest that it is possible to discover causal networks using fuzzy cognitive map learning. We also compared the proposed algorithm with ant colony optimization, differential evolution, and particle swarm optimization in a decomposed framework. Comparison results reveal the advantage of the decomposed genetic algorithm on datasets with small data volumes, large network scales, or the presence of noise.

1 INTRODUCTION

Fuzzy Cognitive Map (FCM) is a graph model that visualizes expert knowledge as a weighted directed graph [1]. The nodes in FCMs represent the concepts to be modeled, and the signed weights between the nodes represent the strength of causal relations. Traditionally, the initial weights are assigned by domain experts. In addition, Hebbian-based learning algorithms [2-4] can be used to resolve conflicts among experts and to improve the accuracy of the weights. The resulting FCM model can be used to study the properties of the complex system under investigation.

There are several advantages in applying FCMs to model complex systems. Because FCMs do not have hidden nodes, the dynamics of the system can be interpreted easily in terms of interactions among a set of well-defined, real-world concepts. Domain experts can also use linguistic terms to define the weights, and the simulation results can thus be easily interpreted in linguistic terms. Furthermore, FCMs can also generate rich dynamics although the models are simple [5]. Due to these advantages, FCMs have been applied to study a wide variety of complex systems, such as, engineering control systems [6-8], on-line design of fuzzy controllers [9-11], situation-aware computing [12], medical decision support systems [13-15], educations [16], and ecosystems [17]. FCMs have also been used together with other techniques to model complex systems. Examples include cellular automata [18], gray system theory [19], and petri nets [20].

In spite of the wide application areas of FCMs, to the best of our knowledge, there is no report on applying FCMs to the causal inference problems. Different from the other applications of FCMs where domain knowledge is applied to construct FCMs, the application of FCMs to causal inference may help discover new domain knowledge.

Causal inference is an important problem which mainly focuses on discovering plausible causal relations between the concepts or nodes. An example of causal inference problem is the reverse engineering of gene regulatory networks (GRNs). GRNs consist of genes and their interaction relations. The expression level of a gene may cause an increase (activation) or decrease (repression) in the expression level of another gene. These causal relations among the genes are critical to understanding the functions of the cells [21]. However, many of these relations are unknown to domain experts and therefore causal inference algorithms are needed to discover these relations.

Although many methods have been developed to infer GRNs from gene expression data [22], we believe FCMs could be used to better represent and discover the relations in GRNs. The most widely used method in the literature include Boolean networks [23, 24], Bayesian networks [25], dynamic Bayesian networks [26], ordinary differential equations [27, 28], and correlation and mutual information based methods [29, 30]. These methods represent the gene expression levels using either Boolean values or scaled real values. Applying FCMs to GRN inference problems could provide a good balance between the Boolean and real value representation methods and could potentially outperform the existing methods.

In this paper, we propose to apply FCM learning algorithms to construct GRNs from data. Although there are existing FCM learning algorithms that could be applied to construct FCMs from data, there are several concerns need to be addressed, including whether the existing FCM learning framework could be applied to causal inference problem, which algorithm performs the best for this particular problem, and how to learn FCMs with a large number of nodes. These three concerns are further discussed in the following.

(1) *The applicability of FCM learning algorithms to the reverse engineering of complex causal networks.* Research advances in FCM learning algorithms provide the opportunity to discover knowledge from historical data; however, to the best of our knowledge, there is no report on the reverse engineering of causal networks using FCMs. Most of the studies apply data-driven FCM learning algorithms to problems focused on minimizing the difference between an output sequence and historical data, i.e., the simulation error [31-33]. However, reverse engineering of causal networks requires an accurate estimation of the weights, i.e., minimizing the model error, which is not directly related to simulation error. Usually, there are many different FCMs with the same or very similar simulation results. The implications of this for the discovery of causal networks are not well-understood. Several studies have assessed the accuracy of weights in the learned FCMs [34, 35]. However, the accuracy of the FCMs is relatively low and it is unknown if there is a significant correlation between the accuracy of simulation result and the accuracy of the structures and weights of the FCMs. The *first aim* of this paper is to evaluate the FCM learning algorithms using structure-based performance measures and determine if simulation accuracy can be used as a reliable objective function in reverse engineering applications.

(2) *The choice of optimization algorithms*. Although a large number of meta-heuristic FCM learning algorithms exist and several reviews [33, 36] can be found in the literature, there are very few comprehensive comparisons [37]. Furthermore, the best algorithm for one application area may not be the best for another, including the application and objective function used in this paper. The *second aim* of this paper is to compare the performance of several widely used algorithms, including an ant colony optimization algorithm for real parameters (ACOR) [38], a differential evolution (DE) algorithm [39], a particle swarm optimization (PSO) algorithm [40, 41], and a variant of real-coded genetic algorithm (RCGA) proposed in this paper.

TABLE I
SIZE OF FCMS (MAXIMUM NUMBER OF NODES) IN PREVIOUS STUDIES

Reference	Year	Algorithm	Size
Koulouriotis et al. [42]	2001	Evolutionary strategies	6
Parsopoulos et al. [43]	2003	PSO	5
Papageorgiou et al. [44]	2005	Hybrid DE and nonlinear HL	8
Papageorgiou et al. [45]	2006	Active HL / nonlinear HL	5
Ghazanfari et al. [46]	2007	Hybrid RCGA and SA	15
Stach et al. [47]	2007	Parallel RCGA	80

Stach et al. [48]	2008	Data-driven nonlinear HL	20
Alizadeh et al. [49]	2009	Chaotic SA	14
Petalas et al. [37]	2009	Memetic PSO	18
Stach et al. [32]	2010	Divide and conquer RCGA	40
Baykasoglu et al. [50]	2011	Extended great deluge	6
Zhang et al. [51]	2011	Gradient residual algorithm	7
Madeiro et al. [52]	2012	RCGA with gradient search	38
Stach et al. [34]	2012	Sparse RCGA	40
Chen et al. [35]	2012	ACO	40
Chen et al. [53]	2012	Decomposed ACOR	100
Huang et al. [54]	2013	Extreme learning machine	6
Yesil et al. [55]	2013	Artificial bee colony	13
Gregor et al. [56]	2013	Gradient-based search	20
Kannappan et al. [57]	2013	Artificial immune systems	26
Napoles et al. [58]	2014	Hybrid PSO and ACO	25
This paper	-	Decomposed RCGA with tournament selection	300

ACO, ant colony optimization; ACOR, ACO for real parameters; DE, differential evolution; HL, Hebbian learning; PSO, particle swarm optimization; RCGA, real-coded genetic algorithm; SA, simulated annealing.

(3) *The applicability of the meta-heuristic algorithms to large-scale FCM learning problems*. It is difficult to learn large scale FCMs because the search space grows exponentially with the number of nodes [35]. Most learning algorithms have been applied to FCMs with a small number of nodes as summarized in **Table I**. Only a few studies have tried to learn large scale FCMs. For example, Stach et al. proposed four versions of RCGA to learn FCMs with 40 nodes [32, 34, 59] and 80 nodes [47] respectively. Chen et al. [53] applied an ant colony optimization algorithm to learn FCMs with 100 nodes, but the accuracy was relatively low. For real-world problems, such as gene regulatory network (GRN) inference, there could be more than 100 nodes. The *third aim* of this paper is thus to propose a new FCM learning algorithm based on RCGA with tournament selection [60] and a decomposed problem formulation [53]. The proposed algorithm is applied to large scale FCMs with up to 300 nodes.

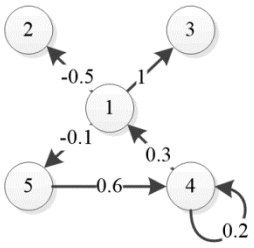
The rest of the paper is organized as follows. Section 2 introduces the necessary background on FCMs, data-driven FCM learning algorithms and the GRN inference problem. Section 3 provides a description of the proposed FCM learning algorithm based on RCGA. Section 4 presents the experimental design. Section 5 presents the results and Section 6 summarizes the main contributions of this paper.

2 BACKGROUND

2.1 Fuzzy Cognitive Maps

Fuzzy cognitive maps (FCMs) were proposed by Kosko as a generalization of cognitive maps [1]. An FCM is a graph with N_N nodes. Every node represents a concept in the system

under investigation. Values ranging [0,1] are assigned to the nodes to represent the status (or activation degrees in FCM terminology) of the nodes. The value of node i is denoted as C_i (i=1, 2, …, N_V). The nodes are connected with directed and weighted edges. The directions of the edges represent the causal effect of nodes on each other. Take the FCM shown in **Fig. 1** as an example. There is an edge from node 1 to node 2, which means that any change in the value of node 1 will cause a change in the value of node 2. The weight of the edge from node i to node j is denoted as $w_{ij}\in[-1,1]$. A positive w_{ij} represents an excitatory relation from node i to node j, i.e., an increase (decrease) in C_i will cause an increase (decrease) in C_j, while a negative w_{ij} represents an inhibitory relation, i.e., an increase (decrease) in C_i will cause a decrease (increase) in C_j. The weight w_{ij} is 0 if there is no causal relation from node i to node j.



(a) Graph Representation

	1	2	3	4	5
1	0	-0.5	1	0	-0.1
2	0	0	0	0	0
3	0	0	0	0	0
4	0.3	0	0	0.2	0
5	0	0	0	0.6	0

(b) Matrix Representation

Fig. 1. An example of a FCM and its equivalent weight matrix.

The dynamics of the node values in the FCMs is simulated by using the following equation:

$$C_i^{(t+1)} = f\left(\sum_{i=1}^{N_V} w_{ij} C_i^{(t)}\right) \tag{1}$$

where $C_i^{(t)}$ is the value of node i in iteration t, $f(\cdot)$ is an activation function that is used to restrict the node values within the range of [0,1]. There are three widely-used activation functions: the signum function, the trivalent function, and the sigmoid function [61]. We use the sigmoid function in this paper because it has been used by a large number of studies on the application of FCMs and by most of the studies on FCM learning algorithms. Furthermore, two comparison studies have suggested that the sigmoid function is better than the other two activation functions in general [61, 62].

The sigmoid activation function is defined as follows:

$$f(x) = \frac{1}{1+e^{-\lambda_0 x}} \tag{2}$$

where λ_0 is a parameter that determines the steepness of the sigmoid function at values around 0. Different λ_0 value may work better for different problems. We propose to learn λ_0 along with the weight matrix for real-world applications. However, for the simulated study based on data generated using FCM, λ_0 is set to 5.0, which is the commonly used value in the literature [34, 35].

2.2 Fuzzy Cognitive Map Learning Algorithms

The traditional approach for constructing FCMs is to use expert knowledge. Several algorithms based on Hebbian, i.e., correlational learning, have been proposed to improve the accuracy of the expert-constructed FCM [3, 4, 45]. However, these algorithms are suitable mainly for refining the weights after an initial weight is determined by domain experts.

Data-driven FCM learning algorithms are capable of learning the weights without domain experts' intervention. Instead of using unsupervised learning rules, these data-driven FCM learning algorithms use optimization algorithms to minimize the difference between reference data sequences and the simulated output data sequences. A number of optimization algorithms have been proposed to learn FCM weights, e.g., genetic algorithms [31, 32, 59], particle swarm optimization [37, 43], differential evolution [44, 63], simulated annealing [46, 49], ant colony optimization [35, 64], and gradient-based optimization algorithms [56]. These algorithms have been applied successfully to learn FCMs without domain experts' intervention for a number of problems, such as time series prediction [65], engineering control problems [37], and classification problems [66-68]. These FCM learning methods mainly differ from each other in the choice of objective functions and optimization procedures.

2.2.1 Objective Functions

For simulation and time series prediction problems, the most widely-used objective function is defined as the difference between the observed data at time t+1 and the simulation results at time t+1 based on the observed data at time t [34, 35, 43]. The observed data and the simulation results are usually considered as points in a high-dimensional space. The difference between these points can be defined using the L_1-norm, L_2-norm, or L_∞-norm. Based on a comparison study of these norms, it is reported that the L_2-norm performs best for FCM learning problems [59]. Because the L_2-norm is differentiable, it is also preferred for gradient-based FCM learning algorithms [56]. In this paper, we term the objective function described above the *one-step simulation error* because the FCM simulation is only performed for one time step.

A decomposed objective function has been proposed to improve the performance of the optimization algorithms based on the one-step simulation error [53]. Another objective function based on multi-step simulation error has also been proposed to improve the prediction accuracy of the FCM models [69].

For engineering control problems, the objective function is usually defined as the distance from the steady-state simulation results to the maximal or minimal acceptable values [43]. This objective function is used in control problems because the aim of a control system is to constrain the status variables within a given interval.

2.2.2 Optimization Algorithms

Several gradient-based approaches and a large number of meta-heuristic algorithms have been proposed to learn FCMs.

Gradient-based methods are usually designed for the L_2-norm simulation error function. The

objective function is minimized starting from an initial solution. Because gradient-based methods are local optimization algorithms, a multi-start strategy or hybrid approaches can significantly improve their accuracy [52].

Meta-heuristic algorithms are widely used to learn FCMs. The advantages of these algorithms include their global optimization capability and their ability to optimize non-differentiable objective functions. These algorithms use intelligent random operators to generate new candidate solutions. Some random operators emphasize the exploration of the whole solution space to search for better solutions, while others emphasize the exploitation of already explored regions in order to improve the current solutions. The algorithms find the best solution by iteratively applying these different operators. The balance between exploration and exploitation is considered to be the key factor that determines the effectiveness of the algorithms [70]. However, it is difficult to find a good balance through theoretical studies. Therefore, experimental studies are used to test if a specific algorithm is suitable for a particular problem.

It is difficult to determine which algorithm works best for FCM learning in general, because most experimental studies have been performed on small scale problems and in different application areas. Ghazanfari et al. [46] compared simulated annealing (SA) and a variant of RCGA with problem-specific operators, and the results suggested that RCGA has better performance on small-scale problems while SA performs better on relatively large-scale problems. However, the experiments were only performed on FCMs with fewer than 15 nodes. Based on a literature search, we have found that only RCGA and ACO have been applied to FCMs with more than 40 nodes [34, 35, 47, 53]. Petalas et al. [37] compared PSO, DE and RCGA on several FCM learning problems and the results suggested that PSO performs better than the other two algorithms. However, the comparison was performed on four FCMs with predefined structure, the maximal number of nodes was only 18 and the number of variables to be optimized was only 40, because of the predefined network structure. The scale of this problem is small compared to the other studies listed in **Table I**. These facts motivated us to perform a comprehensive comparison of learning algorithms on large-scale problems and on the particular application area proposed in this paper.

2.3 Gene Regulatory Networks

A GRN consists of a number of nodes and directed edges between the nodes. Every node represents a gene and the directed edges represent causal relations between the genes.

Fig. 2 shows an example of GRN, which is used to test the proposed algorithm in the following sections. It consists of eight genes involved in the DNA SOS repair pathway [71]. It can be used to analyze the activity of the genes. The expression of recA senses DNA damage (which is a complex process not shown in **Fig. 2**). When DNA damage happens, the product of recA, designated RecA, represses the expression of lexA. The gene lexA represses six downstream genes, including polB, umuD, uvrD, uvrA, uvrY, and ruvA. This means when lexA is repressed, these genes will be activated. They will further lead to other interactions to repair the DNA. This GRN, therefore, provides critical insight into how cells respond to DNA damage.

The major challenge in this type of analysis is to discover the GRNs based purely on observed gene expression data (time-series in this paper).

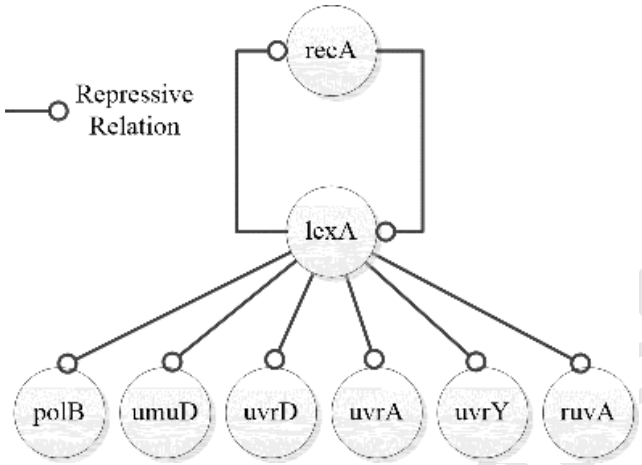


Fig. 2. An example GRN from the *E. coli* SOS network.

As we show in the above simple analysis, the behavior of the cells can be explained by the high or low expression levels of different genes. In fact, the GRNs can be modeled using Boolean networks and Bayesian networks, which represent the expression levels by discrete status [23, 24, 26]. A more realistic model is to use ordinary differential equations, which represent continuous-valued interactions between the genes. However, the nonlinearity of the relationships requires the use of nonlinear ODEs, which are more difficult to infer accurately.

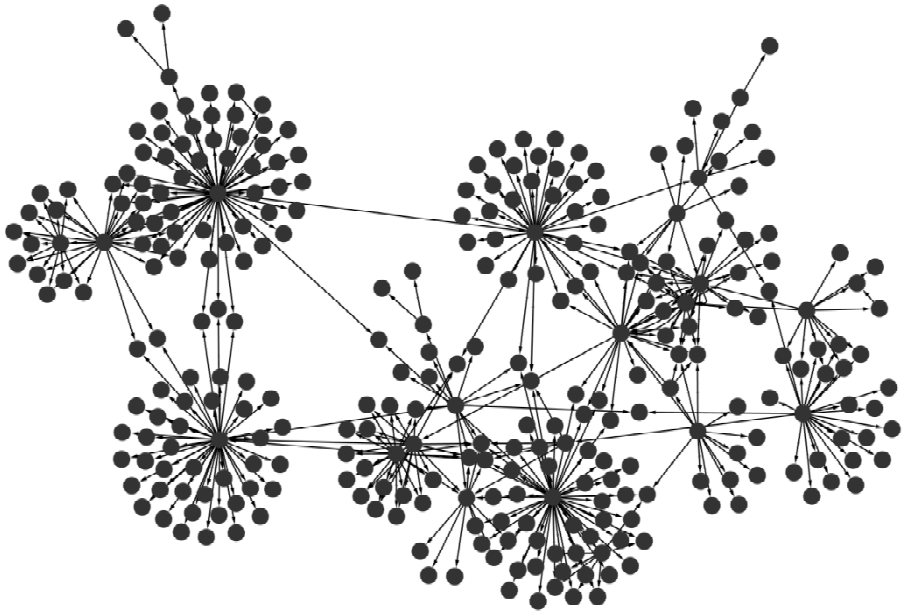


Fig. 3. A GRN from the *E. coli* with 300 genes and 439 regulatory relations. It is one of the GRNs we use to test our algorithm.

Furthermore, the GRNs often contain a large number of nodes, which may further increase the difficulty in constructing and interpreting the model. **Fig. 3** shows an example of a GRN with 300 genes (nodes) and 439 regulatory relations (edges). It is only a part of the even larger E. coli's GRN, which contains more than 4000 genes. And the edges only represent the relations the scientists discovered so far. There may be more relations to be discovered in this GRN. It's important to represent the GRNs in an appropriate abstraction level and develop an efficient algorithm to construct the models from data. In this paper, we propose a new way of representing GRNs by using FCMs and develop a new FCM learning algorithm to efficiently construct the FCM models.

3 PROPOSED APPROACH

As discussed in the previous sections, FCMs could be a better model for GRNs. However, most GRNs consist of too many genes to be constructed by the existing algorithms. In this section, we propose an algorithm that can learn FCMs with hundreds of nodes so that it can be applied to construct large GRNs.

The overview of the proposed approach is shown in **Fig. 4**. The proposed approach consists of two main components, e.g., (1) a decomposed problem formulation [53] with a sparseness penalty, and (2) a real-coded genetic algorithm (RCGA) with tournament selection [60], single point crossover and non-uniform mutation [72]. Furthermore, three other major meta-heuristic optimization algorithms, e.g., an ant colony optimization for real parameters [38], the canonical particle swarm optimization [40], and a differential evolution algorithm [39], are implemented under the same decomposed framework for the purposes of comparison.

In the following, we first introduce the decomposed problem formulation and then describe the proposed optimization algorithms.

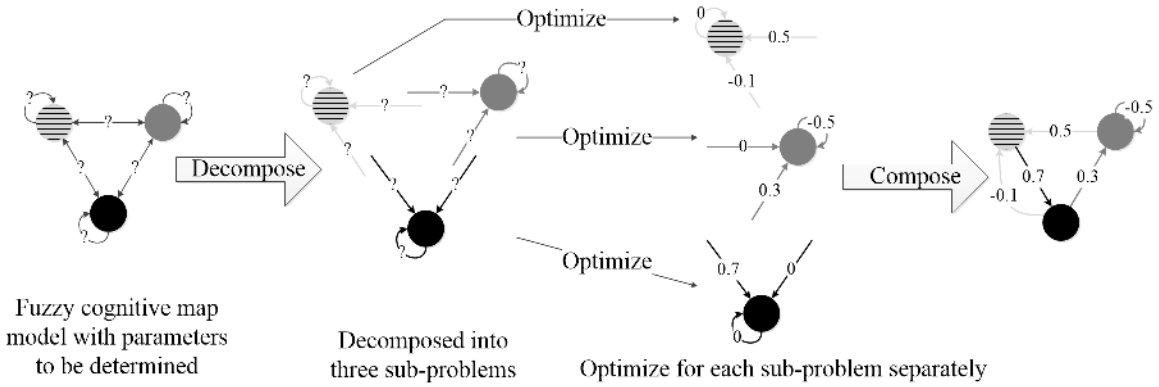


Fig. 4. The framework of the decomposed approach.

3.1 Problem Formulation

The proposed algorithm learns FCMs from time series. Besides the weights matrix, we propose to learn the λ_0 in the transfer function for every node as well. The objective of the

algorithm is to find an optimal FCM that can generate time series as similar to the observed (reference) time series as possible. A sparseness penalty is also included in the objective function because the connections among concepts in real-world applications, including GRNs, are sparse [34, 73]. The objective function before decomposition is:

$$E = \frac{1}{N_{\mathrm{V}}(N_{\mathrm{T}}-1)N_{\mathrm{S}}} \sum_{\substack{2\le t\le N_{\mathrm{T}} \\ 1\le n\le N_{\mathrm{V}} \\ 1\le s\le N_{\mathrm{S}}}} \left(C_n(s,t)-\hat{C}_n(s,t)\right)^2 + p_{\mathrm{S}} \sum_{1\le i,j\le N_{\mathrm{V}}} \left|w_{ij}\right| \tag{3}$$

where N_{V} is the number of nodes in the FCM, N_{S} is the number of time series, N_{T} is the number of time points in every time series, p_{S} is the sparseness penalty factor, $C_n(s,t)$ is the observed data for node n in time series s at time point t, $\hat{C}_n(s,t)$ is the simulation result for node n in time series s at time point t defined as follows.

$$\hat{C}_n(s,t) = f_n\left(\sum_{i=1}^{N_{\mathrm{V}}} w_{in} C_n(s,t-1)\right) \tag{4}$$

where $f_n(\cdot)$ is the activation function for node n:

$$f_n(x) = \frac{1}{1+e^{-\lambda_n x}} \tag{5}$$

where λ_n is the sigmoid parameter for node n. Different parameter values are used for each node to model differences in their sensitivity to changes in their inputs.

The calculation of $\hat{C}_n(s,t)$ only depends on λ_n and the n-th column of the weight matrix $\mathbf{W}_n$ as defined in the following.

$$\mathbf{W}_n = \left[w_{1n}, w_{2n}, ..., w_{N_{\mathrm{N}},n}\right]^{\mathrm{T}} \tag{6}$$

Therefore the objective function E can be formulated as the summation of N_{N} error terms E_n:

$$E_n = \frac{1}{(N_{\mathrm{T}}-1)N_{\mathrm{S}}} \sum_{t=2}^{N_{\mathrm{T}}} \sum_{s=1}^{N_{\mathrm{S}}} \left(C_n(s,t)-\hat{C}_n(s,t)\right)^2 + p_{\mathrm{S}} \sum_{i=1}^{N_{\mathrm{N}}} \left|w_{in}\right| \tag{7}$$

The optimization algorithms are used to optimize λ_n and $\mathbf{W}_n$ separately for every n from 1 to N_{N}. The final result is obtained by directly concatenating the individual results together:

$$\mathbf{W} = \left[\mathbf{W}_1, \mathbf{W}_2, ..., \mathbf{W}_{N_{\mathrm{V}}}\right] \tag{8}$$

$$\boldsymbol{\lambda} = \left[\lambda_1, \lambda_2, ..., \lambda_{N_{\mathrm{V}}}\right] \tag{9}$$

Based on this decomposed approach, the number of variables to be optimized in every single optimization process is reduced from $N_{\mathrm{V}}(N_{\mathrm{V}}+1)$ to $N_{\mathrm{V}}+1$. This is one of the main factors that enable us to learn large scale FCMs.

3.2 Optimization Algorithms

Since the decomposed approach applies the optimization algorithm separately for every node n, the process is described only for one node in this section.

The pseudo-code for $\mathrm{RCGA_D}$ is listed in **Fig. 5**. The major steps in $\mathrm{RCGA_D}$ are described in the following.

Algorithm 1: Real-coded genetic algorithm for one node

Input: observed time series.
Output: the best weights for the incoming edges of the current node and the parameter λ_n for the current node.
Steps:

```
Initialize the population;
Evaluate current population;
while Maximum number of iteration is not reached do
    for id = 1 to population size, stepsize = 2 do
        Select two individuals from the current population
        if a randomly generated number < crossover rate
            Apply crossover operator to the selected individuals and
            insert the two new individuals into the new population;
        else
            Directly copy the selected individuals into the new
            population;
        end if
        Mutate the two new individuals;
    end for
    Evaluate the new population;
    Set the new population as the current population;
end while
Report the results
```

Fig. 5. Pseudo-code for $RCGA_D$

3.2.1 Initialization of the Population

A population of solutions is generated randomly within the variable boundaries according to uniform distribution. In RCGA, the j-th solution is denoted as

$$\mathbf{x}_j = \left[x_{j,1}, x_{j,2}, ..., x_{j,N_V+1} \right] = \left[w_{1n}, w_{2n}, ..., w_{N_V,n}, \lambda_n \right] \tag{10}$$

where n is the current node the $RCGA_D$ is optimizing for.

The interval for the weights are [-1,1]. The interval for the parameter λ_n is $[\lambda_{min}, \lambda_{max}]$, where λ_{min} and λ_{max} are parameters. The choice of these parameters depends on the datasets and the purpose of the experiments. It is described in Section IV.

3.2.2 Evaluating the Population

The weights and λ_n for the current node are first assigned based on (10). Equation (7) is then used to calculate the objective function for every solution in the population. The best solution is also identified in this step.

3.2.3 Selection

Tournament selection [60] is used in $RCGA_D$. Usually, k individual solutions are randomly chosen from the population. The one with the smaller objective function value is selected to perform the subsequent operation. In this paper, we choose k=2 based on experiment results with different k values.

3.2.4 Crossover

A single-point crossover is used in the proposed algorithm [72]. The crossover location l is chosen randomly. The two selected individual solutions $\mathbf{x}_1$ and $\mathbf{x}_2$ can be recombined into two new individuals as follows:

$$\mathbf{x}_1^{(\text{new})} = \left[x_{1,1}, \ldots, x_{1,l}, x_{2,l+1}, \ldots, x_{2,N_V+1} \right] \tag{11}$$

$$\mathbf{x}_2^{(\text{new})} = \left[x_{2,1}, \ldots, x_{2,l}, x_{1,l+1}, \ldots, x_{1,N_V+1} \right] \tag{12}$$

3.2.5 Mutation

A non-uniform mutation is used in the proposed algorithm. Every individual is subjected to mutation. For any specific individual solution $\mathbf{x}_m$, every element in $\mathbf{x}_m$ has a probability p_m of being mutated. The following equation is used to generate a new value for any element that is selected for mutation.

$$x_{m,i}^{(new)} = \begin{cases} x_{m,i} + \left(x_{\max,i} - x_{m,i} \right) \times \delta, & r < 0.5 \\ x_{m,i} - \left(x_{m,i} - x_{\min,i} \right) \times \delta, & r \geq 0.5 \end{cases} \tag{13}$$

where r is a random number in the range of [0,1], i is the index of the element that is selected for mutation, $x_{\max,i}$ and $x_{\min,i}$ are the upper and lower boundaries of the i-th element, δ is defined as follows.

$$\delta = 1 - r_2^{\left(1 - n_\text{iter}/n_\text{maxiter}\right)^b} \tag{14}$$

where r_2 is a newly generated random number in range [0,1], n_iter is the current number of iteration, n_maxiter is the maximum number of iteration, b is a parameter for the mutation.

4 EXPERIMENTAL DESIGN

To demonstrate the capability of the proposed FCM learning algorithm, we test the algorithms on several datasets ranging from simulated data to real-world observation data. The overall experimental steps are shown in **Fig. 6.** In this section, we describe the datasets used to test the algorithms, present the performance measures and lastly discuss the parameter settings for the experiments.

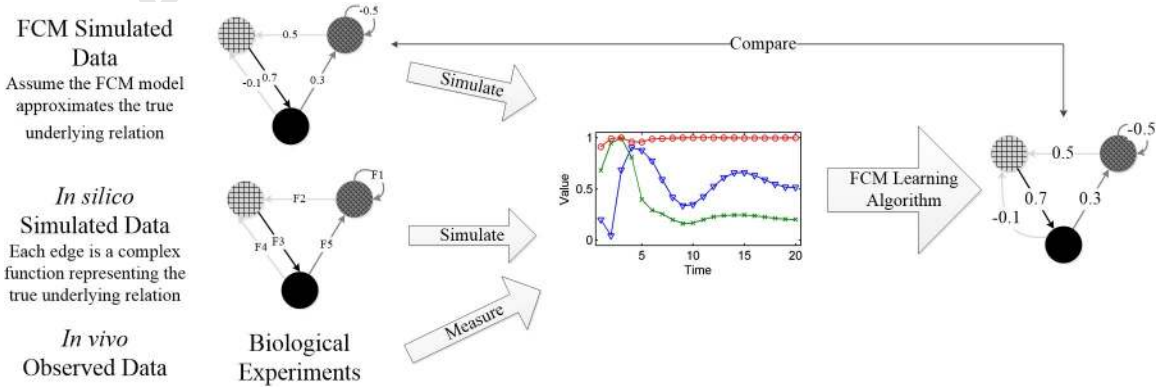


Fig. 6. An overview of the experimental design.

4.1 Datasets

Three types of data are used to test the FCM learning algorithms as summarized in **Table II**.

TABLE II
SUMMARY OF DATASETS

Data Type	#Nets	# Nodes	#Edges	Map Density	Source of Network Structure	Data Length [1]
FCM-simulated	5	10	25, 26, 25, 23, 22 [2]	22%-26%	DREAM4: 2 *E. coli* GRNs, 3 yeast GRNs	1×10, 5×10, 20×10
FCM-simulated	5	50	112, 132, 127, 210, 223 [2]	4.5%-8.9%	DREAM3: 2 *E. coli* GRNs, 3 yeast GRNs [3]	1×10, 5×10, 20×10
FCM-simulated	5	100	276, 349, 295, 311, 293 [2]	2.8%-3.5%	DREAM4: 2 *E. coli* GRNs, 3 yeast GRNs	1×10, 5×10, 20×10
In silico	5	10	15, 16, 15, 13, 12	12%-16%	DREAM4: 2 *E. coli* GRNs, 3 yeast GRNs	5×11, 20×11 [4]
In silico	5	100	176, 249, 195, 211, 193	1.8%-2.5%	DREAM4: 2 *E. coli* GRNs, 3 yeast GRNs	10×11
In silico	1	200	265	0.7%	*E. coli* GRN, extracted using GeneNetWeaver	10×11
In silico	1	300	439	0.5%	*E. coli* GRN, extracted using GeneNetWeaver	20×11
In vivo	1	8	8	12.5%	*E. coli* SOS network	4×49

[1] Data length is presented in the form of "number of time series × number of time points per time series."
[2] Self-cycles are added to these networks. Therefore the number of edges is more than their *in silico* counterparts.
[3] 50-node networks from DREAM3 are used to generate time series, because 50-node networks are not available in DREAM4.
[4] The 5×11 time series come from the DREAM4; the 20×11 time series are simulated using GeneNetWeaver based on network structures in DREAM4.

We designed the FCM-simulated datasets to test whether the FCM learning algorithm is able to discover the true model that generated the time series. In order to reflect real-world scenarios, the directions of influence among genes in the FCMs correspond to the actual directions of influence in the real gene regulatory networks (GRNs) according to the DREAM3 dataset [74, 75]. However, the weight values for the GRN connections are not known, and the weights for the target FCMs are generated randomly using a uniform distribution within the range of $[-1,-0.05]\cup[0.05,1]$. The values between −0.05 and 0.05 are not used, because the weights are too small to cause enough variation in the values of the other nodes. A similar method for generating random weights was also used in several recent studies [34, 35]. The initial states for every node are also generated at random. The time series used for training are obtained by simulating the target networks using Eq. (1).

In silico datasets [76] are used to test the performance of the FCMs on GRN inference problems. It is realistic simulation data based on stochastic differential equations using the GeneNetWeaver software [77]. The network structures are extracted from known GRNs. The simulation model takes two different types of noise into consideration: gene transcription noise

and experimental noise. Compared to FCM-simulated data, the *in silico* datasets are generated based on more realistic simulations and better predict the potential performance of GRN inference algorithms on real data. However, in contrast to real data from unknown GRNs, the structures of the GRNs for the *in silico* datasets are known. For most real gene expression data from real GRNs, especially the large ones, there are many unstudied connections among the genes, making the ground truth unknowable. Therefore, they are not suitable for comparing different algorithms.

The *in silico* datasets used in this paper come from two sources. The 10-node networks with 5×11 time series and the 100-node networks with 10×11 time series come from an annual international competition named the Dialogue for Reverse Engineering Assessments and Methods (DREAM). The simulation is performed by the organizer of DREAM. We use data from DREAM4, which was held in 2010, because a paper compared several GRN inference algorithms on DREAM4 and it allows us to compare with these algorithms on these datasets. DREAM4 includes several types of data but only the time series data is used in this paper.

Another set of *in silico* datasets is generated by us using the GeneNetWeaver software. The 10-node networks with 20×11 time series was used to test the algorithms' performance on datasets with large data volume. The 200-node and 300-node networks were used to test the performance of the algorithms on large networks.

The *in vivo* dataset contains gene expression data measured from the *E. coli* SOS network [71]. It represents a small size GRN that is well studied. Therefore the true network structure is known. Because the gene expression values change vary fast, we compressed the range of the values from [0,1] to [0.25, 0.75]. In this way, the difference between gene expression values at two adjacent time points is smaller.

4.2 Performance Measures

Different performance measures are used for different datasets. The objective function, as a measure of similarity between the observed time series and the simulation results, is used in all datasets. For FCM-simulated datasets, we compared the performance of different algorithms based on model error, although area under the receiver operating characteristics (AUC) and area under the precision recall curve (AUPR) are also calculated. This is because model error is more sensitive to small differences between the true weight matrix and the inferred weight matrix. For *in silico* and *in vivo* datasets, AUC and AUPR are used. Model error is not used because the true weight matrix is not available. The two types of performance measures are defined below:

4.2.1 Performance Based on Weight Errors (*Model Error*)

Model error is the difference between the true weight matrix and the weight matrix of the FCM model learned from data. It is defined as follows.

$$E_{\text{model}} = \frac{1}{N_{\text{V}}^2} \sum_{i=1}^{N_{\text{V}}} \sum_{j=1}^{N_{\text{V}}} \left| w_{ij} - \hat{w}_{ij} \right| \tag{15}$$

where $\hat{w}_{ij}$ is the learned weight from node i to node j.

4.2.2 Performance Based on Binary Networks

Two performance measures based on binary networks are used, i.e., AUC and AUPR [78]. In biomedical studies, the inferred GRNs are usually represented as binary networks so that analysis can be performed to determine the biological meanings of the plausible relations among the genes. The accuracy of the binary network changes as the threshold for generating the binary network from the weight matrix is changed. For the assessment of GRN inference algorithms, we can compare the learned binary networks to the true binary networks. There are four types of outcomes in this comparison: true positive (an edge that exists in the true network and is correctly identified as an existing edge in the learned network), false positive (an edge that does not exist in the true network but is not correctly identified), true negative (an edge that does not exist and is correctly identified), and false negative (an edge that exists but is not correctly identified). True positive rate, false positive rate, recall, and precision are defined based on these four types of outcomes as follows:

$$\text{True Positive Rate} = \text{Recall} = N_{\text{TP}} / \left(N_{\text{TP}} + N_{\text{FN}} \right) \tag{16}$$

$$\text{False Positive Rate} = N_{\text{FP}} / \left(N_{\text{TN}} + N_{\text{FP}} \right) \tag{17}$$

$$\text{Precision} = N_{\text{TP}} / \left(N_{\text{TP}} + N_{\text{FP}} \right) \tag{18}$$

where N_{TP}, N_{FP}, N_{TN}, N_{FN} are number of true positive, false positive, true negative and false negative results respectively.

4.3 Parameter Settings

The parameters for the algorithms are chosen based on experiment results on random networks generated using the same method as described in [35]. In general, grid searches with a step size of 0.1 from 0.1 to 0.9 are performed to determine the best parameter combinations. For the parameters that are outside the range of 0 to 1, they are normalized before parameter search. There are three parameters in RCGA_{D}: the crossover rate and mutation rate are set to 0.6 and 0.3, respectively based on a grid search; and the exponential parameter for the mutation equation is set to 4 based on values suggested in the literature [72]. The parameters for ACOR are set to the same values as in [53]. The parameters for PSO are set to the values used in canonical PSO [40]. Two parameters for DE are determined based on grid search, i.e., the crossover rate and the scale factor. They are set to 0.8 and 0.5, respectively. The population sizes in all four algorithms are directly set to 100 because this value is used in many studies on these algorithms. The maximal number of iterations is 15000. The same value is used in [53] and it is sufficient for the algorithms to converge (based on experimental observations).

The sparseness penalty is set to 0.2 for the large-scale *in silico* datasets with more than 100 (inclusive) nodes. It is set to 0 for the other small-scale *in silico* and *in vivo* datasets.

TABLE III
COMPARISON OF MODEL ERRORS FOR FCM-SIMULATED DATASETS WITHOUT USING SPARSE PENALTY

#Nodes	10	10	10	10	10	50	50	50	50	50	100	100	100	100	100
Baseline	0.517	0.541	0.549	0.538	0.542	0.503	0.504	0.503	0.503	0.503	0.507	0.507	0.509	0.511	0.512
1 Time Series × 10 Time Points															
ACOR	0.208	0.298	0.216	0.290	0.213	0.345	0.354	0.344	0.353	0.351	0.355	0.357	0.356	0.356	0.356
DE	0.281	0.393	0.337	0.408	0.315	0.354	0.354	0.359	0.370	0.366	0.347	0.345	0.344	0.343	0.342
PSO	0.431	0.553	0.468	0.569	0.491	0.616	0.625	0.611	0.613	0.611	0.621	0.625	0.621	0.621	0.624
$RCGA_D$	**0.147**	**0.259**	**0.157**	**0.200**	**0.164**	**0.079**	**0.082**	**0.094**	**0.113**	**0.108**	**0.067**	**0.066**	**0.059**	**0.052**	**0.057**
5 Time Series × 10 Time Points															
ACOR	0.026	0.187	0.072	0.157	**0.115**	0.324	0.339	0.328	0.335	0.330	0.355	0.358	0.357	0.356	0.355
DE	**5E-5**	0.200	**0.043**	**0.128**	0.125	0.355	0.360	0.354	0.362	0.360	0.345	0.348	0.348	0.352	0.350
PSO	0.216	0.430	0.348	0.435	0.351	0.592	0.584	0.590	0.577	0.573	0.596	0.596	0.588	0.583	0.587
$RCGA_D$	0.043	**0.161**	0.093	0.144	0.116	**0.063**	**0.068**	**0.068**	**0.089**	**0.090**	**0.048**	**0.052**	**0.049**	**0.048**	**0.049**
20 Time Series × 10 Time Points															
ACOR	5E-6	0.030	0.002	2E-4	0.004	0.246	0.285	0.267	0.265	0.240	0.348	0.350	0.350	0.345	0.343
DE	**1E-6**	**0.005**	**2E-6**	**4E-6**	**4E-6**	0.266	0.313	0.286	0.285	0.259	0.338	0.341	0.342	0.341	0.333
PSO	0.088	0.207	0.116	0.159	0.136	0.532	0.534	0.527	0.528	0.523	0.555	0.555	0.553	0.552	0.547
$RCGA_D$	0.001	0.068	0.012	0.023	0.025	**0.056**	**0.062**	**0.061**	**0.079**	**0.078**	**0.045**	**0.049**	**0.046**	**0.046**	**0.046**

The best result in each group is shown in bold.

TABLE IV
COMPARISON OF AUCS FOR FCM-SIMULATED DATASETS WITHOUT USING SPARSE PENALTY

#Nodes	10	10	10	10	10	50	50	50	50	50	100	100	100	100	100
Baseline	0.524	0.530	0.482	0.515	0.498	0.500	0.491	0.490	0.495	0.503	0.503	0.494	0.494	0.493	0.504
1 Time Series × 10 Time Points															
ACOR	0.744	0.617	0.704	0.619	0.716	0.518	0.525	0.507	0.523	0.520	0.511	0.510	0.505	0.502	0.514
DE	0.750	0.654	0.607	0.601	0.675	0.522	0.515	0.511	0.512	0.535	0.515	0.503	0.510	0.499	0.508
PSO	0.560	0.551	0.517	0.530	0.544	0.503	0.514	0.500	0.508	0.500	0.499	0.506	0.490	0.500	0.503
$RCGA_D$	**0.766**	**0.665**	**0.780**	**0.715**	**0.748**	**0.562**	**0.551**	**0.618**	**0.596**	**0.589**	**0.552**	**0.545**	**0.591**	**0.565**	**0.542**
5 Time Series × 10 Time Points															
ACOR	0.993	0.763	0.923	0.872	**0.909**	0.541	0.520	0.536	0.553	0.562	0.518	0.516	0.514	0.530	0.532
DE	**1.000**	**0.825**	**0.953**	0.911	0.885	0.544	0.534	0.554	0.570	0.569	0.524	0.523	0.531	0.529	0.515
PSO	0.797	0.578	0.646	0.583	0.677	0.529	0.504	0.513	0.493	0.510	0.500	0.500	0.495	0.500	0.497
$RCGA_D$	0.990	0.705	0.900	**0.932**	0.900	**0.677**	**0.659**	**0.721**	**0.707**	**0.706**	**0.601**	**0.611**	**0.679**	**0.670**	**0.671**
20 Time Series × 10 Time Points															
ACOR	**1.000**	0.991	**1.000**	**1.000**	0.999	0.693	0.642	0.668	0.681	0.725	0.574	0.582	0.598	0.608	0.618
DE	**1.000**	**0.998**	**1.000**	**1.000**	**1.000**	0.688	0.648	0.684	0.697	0.726	0.569	0.571	0.603	0.575	0.587
PSO	0.942	0.798	0.887	0.898	0.889	0.535	0.498	0.520	0.510	0.512	0.500	0.497	0.501	0.514	0.510
$RCGA_D$	**1.000**	0.974	**1.000**	0.999	0.999	**0.886**	**0.860**	**0.866**	**0.860**	**0.875**	**0.826**	**0.786**	**0.871**	**0.828**	**0.856**

The best result in each group is shown in bold.

4 RESULTS AND DISCUSSION

The datasets described above were used to systematically compare the different algorithms. In the following, the performance comparisons on the three types of datasets are discussed first, followed by a discussion of the applicability of FCMs to reverse engineering problems, the applicability of FCMs to large networks, and performance comparison across different datasets.

4.1 FCM-simulated Datasets

FCM-simulated datasets are used to demonstrate that the proposed FCM learning algorithm can reverse engineer the weights for the FCM models. Model errors, AUCs and AUPRs are calculated for the FCM-simulated datasets. Although the AUPRs and the AUCs are different in values, the trends are very similar. Therefore the AUPRs are not shown here. The model errors and AUCs are shown in **Table III** and **Table IV**, respectively. The standard deviations for most

of the average values are one order of magnitude smaller and they are not shown in the tables.

It is observed that $RCGA_D$ outperforms the other algorithms for most of the experiments and PSO performs significantly worse than the other algorithms. However, an interesting observation is that DE performs best on all five 10-node networks when 20 time series are used in the learning process, and DE performs best on only three out of five 10-node networks when 5 time series are used. When only 1 time series is used, DE performs consistently and significantly worse than $RCGA_D$. Similar trends also exist in AUCs, but the difference between different algorithms is smaller than that in model errors. Based on these observations, we conclude that $RCGA_D$ performs consistently well; however, DE performs better if there is a large volume of data available and the scale of the FCM is small.

Another observation is that the model errors for $RCGA_D$ decrease when the scale of the FCM increases. To explain this observation, we calculated the baseline model error for every network. It is calculated by generating random weights and then evaluating the accuracy of the FCMs. For FCMs that are sparser, there are more zero weights. Because the weights range from −1 to 1, the existence of more zero weights results in smaller expected errors. The 50-node networks in the FCM-simulated data are sparser than the 10-node networks, and the 100-node networks are sparser than the 50-node networks. Therefore the model errors decrease slightly along with the decrease of the baseline model error. For the three algorithms other than $RCGA_D$, the model errors increase as the FCM scales increase. This is because these three algorithms are not as scalable as $RCGA_D$ and the model errors increase faster than the decrease of baseline model error. There is also another reason that may lead to the decrease in model errors for $RCGA_D$. The single-point crossover operator is capable of preserving large solution segments. Thus, continuous zero weights in the solutions are better preserved once they are found in $RCGA_D$.

Several other studies have also applied RCGA to learn FCMs [32, 34, 47]. It is difficult to directly compare $RCGA_D$ with the other RCGA variants in the literature, because different datasets were used to test the algorithms. Typically, the studies used randomly generated network structures and the experiments were only performed five times with a different network structure on each trial. This would be expected to result in a large variation in the performance measures. However, several 10-node networks used in [34] have a very similar map density (the ratio of the number of existing edges and the maximum possible number of edges) as the ones used in this paper and we can compare these results. The proposed $RCGA_D$ in this paper is tested on five 10-node networks with map densities ranging from 22% to 26%; while the RCGA in [34] was tested on five random 10-node networks with a map density of 20% and another five random 10-node networks with a map density of 40%. The model errors for RCGA on FCMs with 20% and 40% of map densities are 0.398 and 0.385 respectively; while the average model error for the 5 networks using $RCGA_D$ is 0.185, and the largest model error is only 0.259. The advantage of $RCGA_D$ over RCGA lies mainly in the decomposed framework. As we have compared in a previous study [53], ACOR's performance also improves significantly if a decomposed approach is used. Other factors may also contribute to the higher performance, including the combination of the specific genetic operators used in $RCGA_D$.

TABLE V
COMPARISON OF AUCS AND AUPRS FOR IN SILICO DATASETS FROM DREAM4

	#Nodes	10	10	10	10	10	100	100	100	100	100
AUC	ACOR	0.816±0.003	0.628±0.001	0.588±0.001	0.720±0.004	0.853±0.004	0.728±0.008	0.642±0.013	0.699±0.007	0.681±0.009	0.730±0.007
	DE	**0.819**±0.010	0.627±0.003	0.588±0.008	0.717±0.009	0.855±0.005	0.600±0.020	0.540±0.016	0.611±0.022	0.561±0.013	0.598±0.018
	$RCGA_D$	0.803±0.002	**0.639**±0.001	**0.598**±0.002	**0.739**±0.001	**0.884**±0.001	**0.755**±0.009	**0.656**±0.007	**0.720**±0.007	**0.687**±0.005	**0.738**±0.008
	DBN1 [1]	**0.73**	0.64	0.68	**0.85**	**0.92**	**0.68**	**0.64**	**0.68**	0.66	**0.72**
	DBN2 [1]	**0.73**	**0.66**	**0.77**	0.80	0.84	0.59	0.56	0.59	**0.67**	0.71
	ODE [1]	0.62	0.63	0.58	0.63	0.68	0.55	0.55	0.6	0.54	0.59
AUPR	ACOR	0.481±0.033	0.426±0.009	0.234±0.021	0.493±0.045	0.596±0.082	**0.186**±0.002	0.117±0.001	0.160±0.003	0.143±0.001	0.170±0.001
	DE	**0.490**±0.038	0.419±0.009	0.246±0.031	0.490±0.050	0.600±0.086	0.037±0.008	0.032±0.003	0.074±0.015	0.036±0.007	0.042±0.007
	$RCGA_D$	0.481±0.003	**0.430**±0.005	**0.267**±0.023	**0.536**±0.012	**0.766**±0.010	**0.186**±0.004	**0.128**±0.003	**0.204**±0.005	**0.149**±0.003	**0.190**±0.004
	DBN1 [1]	0.37	0.34	0.45	**0.69**	**0.77**	**0.11**	**0.10**	**0.13**	**0.10**	**0.11**
	DBN2 [1]	**0.38**	**0.41**	**0.49**	0.46	0.64	0.08	0.05	0.11	**0.10**	0.09
	ODE [1]	0.27	0.32	0.21	0.23	0.25	0.02	0.03	0.03	0.02	0.03

The results are presented in the format of average value ± standard deviation. The standard deviations for the deterministic algorithms, e.g., DBN1, DBN2, and ODE are not shown. The best results for FCM algorithms and the best results for the deterministic algorithms are both shown in bold. The best results for all the algorithms are shaded in gray.

[1] The results for DBN1, DBN2 and ODE are reported in [79]. DBN1 is a dynamic Bayesian network approach without hidden nodes. DBN2 is a dynamic Bayesian network approach with hidden nodes.

4.2 In Silico Datasets

While the results from the previous section suggest the proposed algorithm can reverse engineer the weights in the FCMs from simulated data accurately, we further test the algorithms on more realistic simulation data generated from *in silico* models and compare them with other GRN inference algorithms.

Table V reports the performance measures, including AUCs and AUPRs, for FCM models constructed using three different learning algorithms, two DBN variants, and an ODE model [79]. PSO is not evaluated on the *in silico* datasets, because the performance of PSO is significantly worse than the other three algorithms on the FCM-simulated datasets.

$RCGA_D$ performs better than ACOR and DE on 9 out of 10 networks. Comparing the results based on FCM and $RCGA_D$ to the two DBN variants and the ODE, $RCGA_D$ performs better on all five 100-node networks and only one 10-node network. In general, FCM is applicable to the GRN inference problem and FCM in combination with $RCGA_D$ based learning algorithm is more scalable than other approaches for inferring large-scale networks.

Because both FCM and ODE use weighted sum to model the dependence of the gene expression levels in two adjacent time points, it is interesting to compare the accuracies of these two models. It is observed that the FCMs constructed by $RCGA_D$ are more accurate than the ODE models for all ten networks. The FCMs constructed by ACOR and DE are also more accurate than the ODE models for most of the networks. The performance difference between FCM and ODE is mainly due to the use of the sigmoid activation function, because the activation function is the main difference between the two types of models. In FCMs, the sigmoid activation function is used for every node to map the weighted sum into the range [0,1], whereas in ODEs, the weighted sum is directly used as the output of every node. A possible explanation for the benefit of using sigmoid function is that the sigmoid function models the saturation effect

that often exists in real-world scenarios [80], while the nonlinearities it introduces in the optimization process are overcome by the general strength of $RCGA_D$.

4.3 In vivo Datasets

Although realistic simulation dataset provides a good evaluation of the FCM learning algorithms, we further demonstrate the performance of the algorithms on the real data from the *E. coli* 8-gene SOS network. The results are compared in **Fig. 7**. Three FCM learning algorithms, including ACOR, DE and $RCGA_D$, perform better than the other algorithms, including LP, LASSO, PCA-PCC and PCA-CMI [81]. These results suggest that FCM is a better model than the other four models for the inference of gene regulatory networks. It is observed that DE slightly outperforms $RCGA_D$ in this dataset.

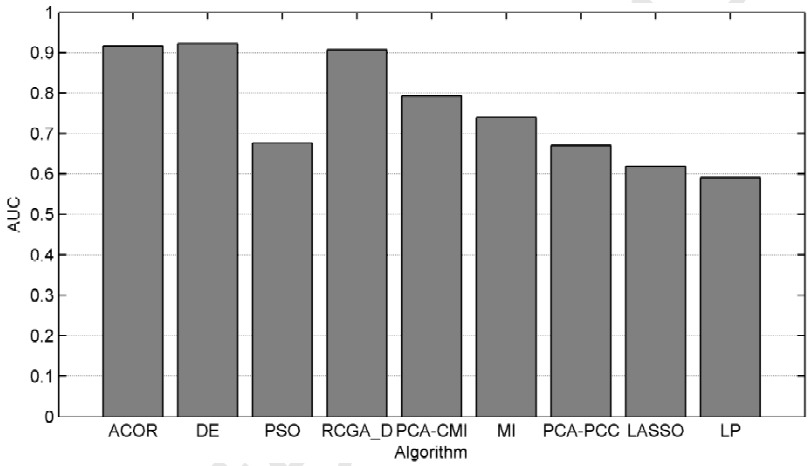


Fig. 7. The performance comparison of gene regulatory network inference algorithms on the E. coli 8-gene SOS network. The results for LP, LASSO, PCA-PCC and PCA-CMI come from [81].

4.4 Applicability of the Proposed Algorithm to Large Networks

To test the applicability of the proposed algorithm on large networks, several GRNs with 200 and 300 nodes were extracted from the known network of *E. coli* using GeneNetWeaver. The results are shown in **Table VI**. The AUCs for the 300-node networks are around 0.7, which is similar to the AUCs for the 100-node networks as shown in **Table VI**. The AUPRs are lower than the results for the FCM learning algorithms in **Table VI**; however, the AUPRs for $RCGA_D$ on 200-node and 300-node problems are higher than the AUPRs for ODE, a widely used method for GRN inference, on smaller problems with only 100 nodes.

TABLE VI
COMPARISON OF THE FCM LEARNING ALGORITHMS ON LARGE NETWORKS

#Nodes	200		300	
	AUC	AUPR	AUC	AUPR
ACOR	**0.699**±0.003	**0.078**±0.001	0.702±0.005	0.052±0.002
DE	0.538±0.022	0.009±0.001	0.531±0.011	0.006±0.001
$RCGA_D$	0.685±0.007	0.038±0.002	**0.749**±0.005	**0.057**±0.003

The results are presented in the form of average value ± standard deviation.

4.5 Comparing the FCM Learning Algorithms across Datasets

From the experimental results on different datasets, we observe that the performance of different FCM learning algorithms varies across different datasets. In general, $RCGA_D$ performs better than the other FCM learning algorithms on the problems with a large number of nodes, small data volume and high noise level. This observation is analyzed in the following.

4.5.1 Performance Comparison on Large Scale Problems

Based on the results in **Table III**, **Table IV** and **Table V**, $RCGA_D$ performs better than the other FCM learning algorithms on 50-node and 100-node FCM learning problems. We may expect that $RCGA_D$ also performs best on the 200-node and 300-node problems; however, $RCGA_D$ performs worse than the other algorithms on the 200-node problem as reported in **Table VI**. The possible cause of this observation is that the 200-node and 300-node networks used in this paper are much sparser than the 100-node networks. The map densities for the 200-node and 300-node networks are smaller than 0.7% and the map densities for the 100-node networks are larger than 1.8%. For highly sparse networks, there are very few cycles and therefore the interaction patterns for the nodes may be simpler. The benefit of $RCGA_D$ is not fully utilized. However, comparing the results for the 200-node and 300-node problems, $RCGA_D$ still performs better than the other algorithms on the larger network. To conclude, $RCGA_D$ performs better on large problems.

4.5.2 Performance Comparison on Small Scale Problems

For the small scale problems (≤10 nodes), $RCGA_D$ performs better on the datasets with small data volume and high noise level. To better demonstrate this observation, a summary of the best performing FCM learning algorithms for different small scale datasets is provided in **Table VII**. The data volumes in the table are sorted from small to large. The FCM-simulated datasets with 5 time series × 10 time points and the *in silico* datasets with 5 time series × 11 time points are considered as having similar volume of data.

TABLE VII
THE BEST PERFORMED ALGORITHMS FOR SMALL SCALE DATASETS

Data Volume	Noise Level (Dataset)		
	None (FCM-simulated)	Low (*in silico*)	High (*in silico* and *in vivo*) [2]
1×10	$RCGA_D$	-	-
5×10/5×11 [1]	DE/ACOR	-	$RCGA_D$
20×10/20×11 [1]	DE/ACOR	$RCGA_D$	$RCGA_D$
5×40	-	-	$RCGA_D$
5×49	-	-	DE/ACOR

This table is based on results on 10-node networks (the FCM-simulated datasets and the *in silico* datasets) and 8-node networks (the *in vivo* dataset).

[1] Data volume for the FCM-simulated datasets is 5×10 or 20×10. Data volume for the *in silico* datasets is 5×11 or 20×11.

[2] The last two results with data volumes of 5×40 and 5×49 are obtained on the *in vivo* datasets. The other two results are obtained on the *in silico* datasets.

The noise levels of the datasets are determined based on the data generation methods and they can also be approximately observed from the time series examples as shown in **Fig. 8**. There is no noise in the FCM-simulated data, because we do not include any noise in the equations used for simulation. It is observed that the lines are smooth in **Fig. 8**(a). The noise level for the *E. coli* SOS network data is considered as high because it is recorded *in vivo*, which contains at least two types of noises, i.e., intrinsic gene regulation noise and experimental (measurement) noise. It is observed from **Fig. 8**(b) that there are some fluctuations starting approximately from the 20-th time point. The noise level for one *in silico* dataset (the 10-node network with 20×11 time series) is considered as low because this dataset is generated using GeneNetWeaver without incorporating experiment noise. The noise level for the other *in silico* datasets is considered as high because it is generated using GeneNetWeaver with both types of noises. The different noise levels of the *in silico* datasets are illustrated in **Fig. 8**(c) – **Fig. 8**(e). Comparing with the data without noise shown in **Fig. 8**(c), the two noisy data contains large variations that is not generated by the underlying gene regulation dynamics. These variations pose a significant problem for the network inference algorithms.

It is observed from **Table VII** that $RCGA_D$ outperforms the other FCM learning algorithms on datasets with small data volume or high noise levels.

TABLE VIII
COMPARISON OF THE AUCS OF THE FCM LEARNING ALGORITHMS ON THE *E. COLI* SOS NETWORK DATASETS WITH DIFFERENT DATA VOLUMES

Data Volume	ACOR	DE	PSO	$RCGA_D$	Diff [1]
5×20	0.869±0.003	**0.873**±0.006	0.667±0.079	0.892±0.002	0.019
5×30	0.911±0.006	**0.916**±0.007	0.680±0.118	0.925±0.002	0.009
5×40	0.944±0.008	**0.952**±0.009	0.757±0.082	0.955±0.003	0.003
5×49	**0.916**±0.006	0.922±0.008	0.676±0.114	0.907±0.004	−0.015

The best AUCs for each dataset are shaded in gray and the second place AUCs are shown in bold.
[1] Diff is defined as the AUC of $RCGA_D$ subtracted by the best AUC of the other three algorithms.

To further demonstrate the advantage of $RCGA_D$ on datasets with small data volume, we generate several different datasets by removing data for some of the time points from the *in vivo E. coli* SOS network dataset. The performance of the FCM learning algorithms on the datasets with different volumes is shown in **Table VIII**. It is observed that $RCGA_D$ outperforms the other algorithms when the number of time points is smaller than 40; however, the performance difference between $RCGA_D$ and the other algorithm decreases as the number of time points increases. The performance of $RCGA_D$ is worse than ACOR and DE when the number of time points is 49.

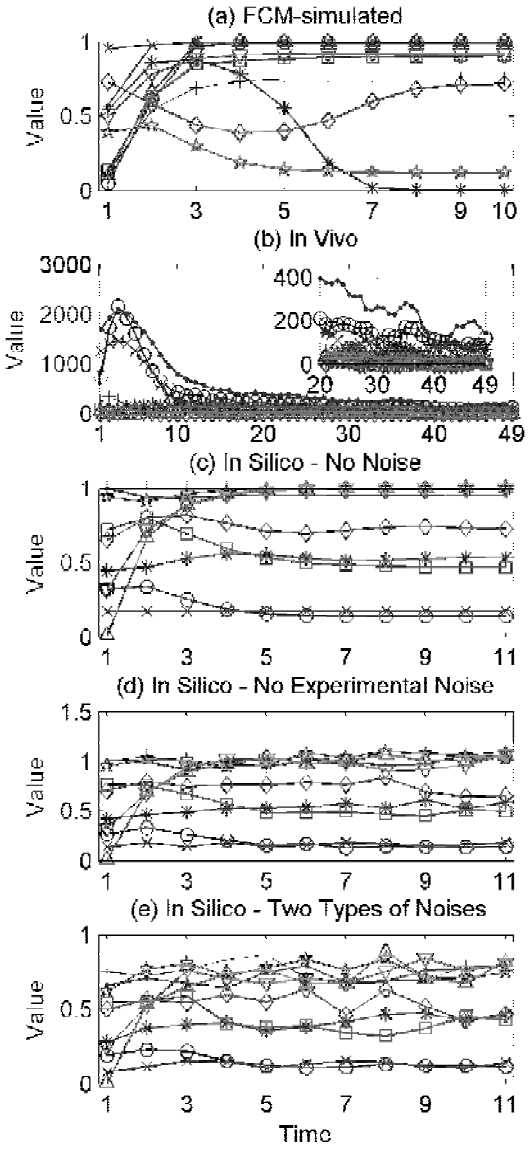


Fig. 8. Example time series from different datasets. The datasets contains 8 to 10 time series each. The time series for each dataset are plotted using different shades of gray and different markers in one subplot. The noise levels in the datasets can be identified qualitatively by visual examination. Especially, since subplots (c) to (e) are generated using the same network, they would have been identical if there was no noise.

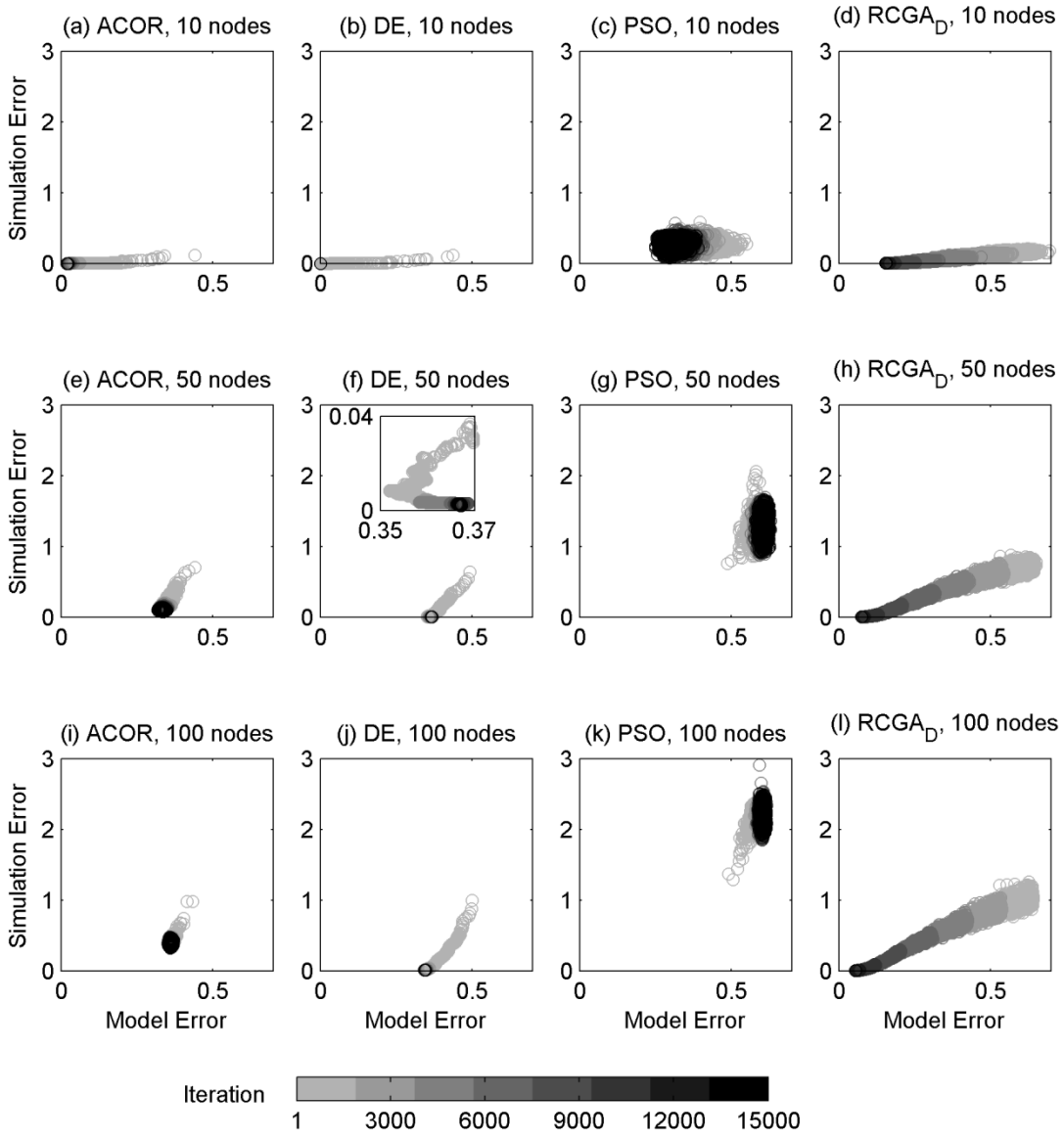


Fig. 9. Scatter plots of simulation errors vs. model errors for the four FCM learning algorithms on FCM-simulated data with five time series. The position of every circle in the subplots represents the simulation error and the model error of the best solution in a different iteration of the optimization algorithm. The different shades of gray represent time step from light to dark. By looking at the circles from light gray to dark gray, we may observe how the two error measures evolve as the optimization algorithms run.

4.6 Applicability of FCMs to Causal Network Reverse Engineering

In this paper, we have demonstrated the application of FCMs to a causal network reverse engineering problem, e.g., the inference of gene regulatory networks. Similar to the algorithms for other reverse engineering problems [82], we optimize the simulation error and expect that the model error will be minimized in this process. The key to the success of this approach is that, for this application, simulation errors are highly correlated with model errors. However, to the best of our knowledge, there is no previous study demonstrating for FCM learning problems.

To evaluate the relation between simulation errors and model errors, the best solutions generated in all trial of the optimization algorithm are recorded. The relation between the simulation errors and the model errors is shown in **Fig. 9**. It is observed that the two error measures are approximately correlated. However, improvement in simulation error does not necessarily lead to an improvement in model error. The most obvious example can be found in **Fig. 9**(b). It shows that when the simulation error drops below 0.01, the model error starts to increase; while for the rest of the simulation error range, model error decreases as simulation error decreases. These observations suggest that simulation error is a good, although not perfect, proxy of model error.

Because we can calculate simulation errors for FCM models generated by different algorithms, one may ask whether it is possible to choose the best model from those generated by these different algorithms based on simulation errors. From the results shown in **Fig. 9**, the answer is negative. It is observed that, for example, the simulation errors for the model generated by DE in the last several trials are much lower than the ones generated by ACOR. If we choose the model based on simulation errors of these two algorithms, we may choose the results from DE, which has a higher model error for this dataset.

Different algorithms have different optimization trajectories as shown in **Fig. 9**. The subplots for $RCGA_D$ are much wider than ACOR and DE. It is possibly because $RCGA_D$ is able to search a wider solution space, which leads to a diverse range of model errors and simulation errors. The searching processes in ACOR and DE are more focused and therefore lead to a narrow distribution of the model errors and simulation errors. The difference in the solution diversity may explain differences in the scalability and robustness of the algorithms. For large scale and noisy datasets, $RCGA_D$ performs better because the candidate solutions generated by $RCGA_D$ are more diverse than ACOR and DE. It prevents the optimization process from becoming trapped in a local optimum. However, it wastes computational resources to explore a wider area of the solution space for datasets with fewer nodes and less noise. In this scenario, ACOR and DE are able to focus on a few prominent areas and therefore perform better than $RCGA_D$.

Comparing the subplots for PSO in **Fig. 9**, it is observed that for the 50-node and 100-node FCM learning problems, the simulation error does not improve during the optimization process and model error increases; while for the 10-node problem, both the simulation error and the model error decrease slightly. This observation suggests that the PSO variant used in this paper is only suitable for small scale FCM learning problems.

5 CONCLUSION

Fuzzy cognitive maps (FCMs) have been applied to a wide variety of problems. However, there is no existing study on applying FCMs to reverse engineer causal networks. In this paper, we focus on developing FCM learning algorithms for the causal network reverse engineering problem. More specifically, we developed an algorithm to construct large gene regulatory networks (GRNs) from gene expression data. The contributions of this paper include:

(1) We explored a new application of FCM learning algorithms, e.g., the reverse engineering

of causal networks from observed time-series. An ideal network reverse engineering algorithm should be capable of optimizing model errors. However, the model errors are not available in the optimization process. In practice, we optimize simulation error instead. Our experiment results show that, for the cases studied in this research, there is a good correlation between simulation errors and model errors, which suggests that the approach we adopted may be broadly applicable.

(2) A new FCM learning algorithm is proposed to learn large scale FCMs. In most of the existing studies, the FCM learning algorithms are only applied to construct FCMs with less than 40 nodes. In this paper, we proposed a new FCM learning algorithm based on a decomposed framework and a real-coded genetic algorithm (RCGA) with tournament selection, the single point crossover operator and the non-uniform mutation are proposed to learn FCMs. The experiments demonstrated that the proposed algorithm is able to learn large scale networks with up to 300 nodes.

(3) A comparison study is performed to demonstrate the advantage and disadvantage of the four optimization algorithms, e.g., the ant colony optimization for real parameters (ACOR), a differential evolution (DE) algorithm, the canonical particle swarm optimization (PSO), the $RCGA_D$ proposed in this paper. The experimental results suggest the proposed $RCGA_D$ outperforms the other algorithms when the data volume is small, network scale is large, or the data has noise.

As this work is a first step towards constructing large scale causal networks, there is ample scope for future work in this area. Possible directions include improving the objective function to reduce the large number of equivalent FCMs, and improving the algorithms with problem-specific operators. Furthermore, the proposed algorithm could be applied to larger gene regulatory networks that are less studied and provide new insight into the cell functions. Finally, the approach may also be applicable to other areas where networks need to be inferred from time-series data, e.g., in models of brain function and social networks.

REFERENCE

[1] B. Kosko, "Fuzzy cognitive maps," *International Journal of Man-Machine Studies,* vol. 24, pp. 65-75, 1986.

[2] G. A. Papakostas, D. E. Koulouriotis, A. S. Polydoros, and V. D. Tourassis, "Towards Hebbian learning of Fuzzy Cognitive Maps in pattern classification problems," *Expert Systems with Applications,* vol. 39, pp. 10620-10629, 9/15/ 2012.

[3] E. Papageorgiou, C. Stylios, and P. Groumpos, "Fuzzy Cognitive Map Learning Based on Nonlinear Hebbian Rule," in *AI 2003: Advances in Artificial Intelligence*, 2003, pp. 256-268.

[4] E. I. Papageorgiou, C. D. Stylios, and P. P. Groumpos, "Active Hebbian learning algorithm to train fuzzy cognitive maps," *International Journal of Approximate Reasoning,* vol. 37, pp. 219-249, 11// 2004.

[5] C. J. K. Knight, D. J. B. Lloyd, and A. S. Penn, "Linear and sigmoidal fuzzy cognitive maps: An analysis of fixed points," *Applied Soft Computing,* vol. 15, pp. 193-202, 2// 2014.

[6] M. Mendonça, B. Angelico, L. V. R. Arruda, and F. Neves Jr, "A dynamic fuzzy cognitive map applied to chemical process supervision," *Engineering Applications of Artificial Intelligence,* vol. 26, pp. 1199-1210, 4// 2013.

[7] C. D. Stylios and P. P. Groumpos, "Fuzzy Cognitive Maps in modeling supervisory control systems," *Journal of Intelligent and Fuzzy Systems,* vol. 8, pp. 83-98, 01/01/ 2000.

[8] C. D. Stylios and P. P. Groumpos, "Modeling complex systems using fuzzy cognitive maps," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on,* vol. 34, pp. 155-162, 2004.

[9] J. L. Gonzalez, O. Castillo, and L. T. Aguilar, "Performance analysis of Cognitive Map-Fuzzy Logic Controller model for adaptive control application," in *Fuzzy Systems, 2008. FUZZ-IEEE 2008. (IEEE World Congress on Computational Intelligence). IEEE International Conference on*, 2008, pp. 2375-2381.

[10] J. L. Gonzalez, L. T. Aguilar, and O. Castillo, "A cognitive map and fuzzy inference engine model for online design and self fine-tuning of fuzzy logic controllers," *International Journal of Intelligent Systems,* vol. 24, pp. 1134-1173, 2009.

[11] J. L. Gonzalez, O. Castillo, and L. T. Aguilar, "FPGA as a Tool for Implementing Non-fixed Structure Fuzzy Logic Controllers," in *Foundations of Computational Intelligence, 2007. FOCI 2007. IEEE Symposium on*, 2007, pp. 523-530.

[12] G. D'Aniello, V. Loia, and F. Orciuoli, "A multi-agent fuzzy consensus model in a Situation Awareness framework," *Applied Soft Computing,* vol. 30, pp. 430-440, 5// 2015.

[13] P. P. Groumpos and A. P. Anninou, "A theoretical mathematical modeling of Parkinson's disease using Fuzzy Cognitive Maps," in *BIBE*, 2012, pp. 677-682.

[14] C. D. Stylios, V. C. Georgopoulos, G. A. Malandraki, and S. Chouliara, "Fuzzy cognitive map architectures for medical decision support systems," *Applied Soft Computing,* vol. 8, pp. 1243-1251, 6// 2008.

[15] E. I. Papageorgiou, C. D. Stylios, and P. P. Groumpos, "An integrated two-level hierarchical system for decision making in radiation therapy based on fuzzy cognitive maps," *Biomedical Engineering, IEEE Transactions on,* vol. 50, pp. 1326-1339, 2003.

[16] S. Hossain and L. Brooks, "Fuzzy cognitive map modelling educational software adoption," *Computers & Education,* vol. 51, pp. 1569-1588, 12// 2008.

[17] K. Kok, "The potential of Fuzzy Cognitive Maps for semi-quantitative scenario development, with an example from Brazil," *Global Environmental Change,* vol. 19, pp. 122-133, 2// 2009.

[18] V. K. Mago, L. Bakker, E. I. Papageorgiou, A. Alimadad, P. Borwein, and V. Dabbaghian, "Fuzzy cognitive maps and cellular automata: An evolutionary approach for social systems modelling," *Applied Soft Computing,* vol. 12, pp. 3771-3784, 12// 2012.

[19] J. L. Salmeron and E. Gutierrez, "Fuzzy Grey Cognitive Maps in reliability engineering," *Applied Soft Computing,* vol. 12, pp. 3818-3824, 12// 2012.

[20] G. Kyriakarakos, A. I. Dounis, K. G. Arvanitis, and G. Papadakis, "A fuzzy cognitive maps–petri nets energy management system for autonomous polygeneration microgrids," *Applied Soft Computing,* vol. 12, pp. 3785-3797, 12// 2012.

[21] G. Karlebach and R. Shamir, "Modelling and analysis of gene regulatory networks," *Nat Rev Mol Cell Biol,* vol. 9, pp. 770-780, 10//print 2008.

[22] D. Marbach, J. C. Costello, R. Kuffner, N. M. Vega, R. J. Prill, D. M. Camacho*, et al.*, "Wisdom of crowds for robust gene network inference," *Nat Meth,* vol. 9, pp. 796-804, 08//print 2012.

[23] I. Shmulevich, E. R. Dougherty, S. Kim, and W. Zhang, "Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks," *Bioinformatics,* vol. 18, pp. 261-274, February 1, 2002 2002.

[24] H. Lähdesmäki, I. Shmulevich, and O. Yli-Harja, "On Learning Gene Regulatory Networks Under the Boolean Network Model," *Machine Learning,* vol. 52, pp. 147-167, 2003/07/01 2003.

[25] S. Imoto, S. Miyano, and T. Goto, "Estimation of Genetic Networks and Functional Structures Between Genes by Using Bayesian Networks and Nonparametric Regression," in *Pacific Symposium on Biocomputing*, 2002, pp. 175-186.

[26] B.-E. Perrin, L. Ralaivola, A. Mazurie, S. Bottani, J. Mallet, and F. d'Alché–Buc, "Gene networks inference using dynamic Bayesian networks," *Bioinformatics,* vol. 19, pp. ii138-ii148, September 27, 2003 2003.

[27] T. S. Gardner, D. di Bernardo, D. Lorenz, and J. J. Collins, "Inferring Genetic Networks and Identifying Compound Mode of Action via Expression Profiling," *Science,* vol. 301, pp. 102-105, July 4, 2003 2003.

[28] M. Bansal and D. D. Bernardo, "Inference of gene networks from temporal gene expression profiles," *Systems Biology, IET,* vol. 1, pp. 306-312, 2007.

[29] A. J. Butte and I. S. Kohane, "Mutual information relevance networks: functional genomic clustering using pairwise entropy measurements," in *Pac Symp Biocomput*, 2000, pp. 418-429.

[30] A. V. Werhli, M. Grzegorczyk, and D. Husmeier, "Comparative evaluation of reverse engineering gene regulatory networks with relevance networks, graphical gaussian models and bayesian networks," *Bioinformatics,* vol. 22, pp. 2523-2531, Oct. 2006.

[31] W. Stach, L. Kurgan, W. Pedrycz, and M. Reformat, "Evolutionary Development of Fuzzy Cognitive Maps," in *Fuzzy Systems, 2005. FUZZ '05. The 14th IEEE International Conference on*, 2005, pp. 619-624.

[32] W. Stach, L. Kurgan, and W. Pedrycz, "A divide and conquer method for learning large Fuzzy Cognitive Maps," *Fuzzy Sets and Systems,* vol. 161, pp. 2515-2532, 10/1/ 2010.

[33] E. I. Papageorgiou, "Learning Algorithms for Fuzzy Cognitive Maps - A Review Study," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 42, pp. 150-163, 2012.

[34] W. Stach, W. Pedrycz, and L. A. Kurgan, "Learning of Fuzzy Cognitive Maps Using Density Estimate," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on,* vol. 42, pp. 900-912, 2012.

[35] Y. Chen, L. Mazlack, and L. Lu, "Learning fuzzy cognitive maps from data by ant colony optimization," in *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference*, Philadelphia, Pennsylvania, USA, 2012, pp. 9-16.

[36] E. I. Papageorgiou and J. L. Salmeron, "A Review of Fuzzy Cognitive Maps Research During the Last Decade," *Fuzzy Systems, IEEE Transactions on,* vol. 21, pp. 66-79, 2013.

[37] Y. G. Petalas, K. E. Parsopoulos, and M. N. Vrahatis, "Improving fuzzy cognitive maps learning through memetic particle swarm optimization," *Soft Computing,* vol. 13, pp. 77-94, 2009/01/01 2009.

[38] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of Operational Research,* vol. 185, pp. 1155-1173, 3/16/ 2008.

[39] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-Art," *Evolutionary Computation, IEEE Transactions on,* vol. 15, pp. 4-31, 2011.

[40] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization," *Swarm Intelligence,* vol. 1, pp. 33-57, 2007/06/01 2007.

[41] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, 1995, pp. 39-43.

[42] D. E. Koulouriotis, I. E. Diakoulakis, and D. M. Emiris, "Learning fuzzy cognitive maps using evolution strategies: a novel schema for modeling and simulating high-level behavior," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, 2001, pp. 364-371 vol. 1.

[43] K. E. Parsopoulos, E. I. Papageorgiou, P. P. Groumpos, and M. N. Vrahatis, "A first study of fuzzy cognitive maps learning using particle swarm optimization," in *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*, 2003, pp. 1440-1447 Vol.2.

[44] E. I. Papageorgiou and P. P. Groumpos, "A new hybrid method using evolutionary algorithms to train Fuzzy Cognitive Maps," *Applied Soft Computing,* vol. 5, pp. 409-431, 7// 2005.

[45] E. I. Papageorgiou, C. Stylios, and P. P. Groumpos, "Unsupervised learning techniques for fine-tuning fuzzy cognitive map causal links," *International Journal of Human-Computer Studies,* vol. 64, pp. 727-743, 8// 2006.

[46] M. Ghazanfari, S. Alizadeh, M. Fathian, and D. E. Koulouriotis, "Comparing simulated annealing and genetic algorithm in learning FCM," *Applied Mathematics and Computation,* vol. 192, pp. 56-68, 9/1/ 2007.

[47] W. Stach, L. Kurgan, and W. Pedrycz, "Parallel Learning of Large Fuzzy Cognitive Maps," in *Neural Networks, 2007. IJCNN 2007. International Joint Conference on*, 2007, pp. 1584-1589.

[48] W. Stach, L. Kurgan, and W. Pedrycz, "Data-driven Nonlinear Hebbian Learning method for Fuzzy Cognitive Maps," in *Fuzzy Systems, 2008. FUZZ-IEEE 2008. (IEEE World Congress on Computational Intelligence). IEEE International Conference on*, 2008, pp. 1975-1981.

[49] S. Alizadeh and M. Ghazanfari, "Learning FCM by chaotic simulated annealing," *Chaos, Solitons & Fractals,* vol. 41, pp. 1182-1190, 8/15/ 2009.

[50] A. Baykasoglu, Z. D. U. Durmusoglu, and V. Kaplanoglu, "Training Fuzzy Cognitive Maps via Extended Great Deluge Algorithm with applications," *Computers in Industry,* vol. 62, pp. 187-195, 2// 2011.

[51] H. Zhang, Z. Shen, and C. Miao, "Train Fuzzy Cognitive Maps by gradient residual algorithm," in *Fuzzy Systems (FUZZ), 2011 IEEE International Conference on*, 2011, pp. 1815-1821.

[52] S. S. Madeiro and F. J. V. Zuben, "Gradient-Based Algorithms for the Automatic Construction of Fuzzy Cognitive Maps," in *Machine Learning and Applications (ICMLA), 2012 11th International Conference on*, 2012, pp. 344-349.

[53] Y. Chen, L. J. Mazlack, and L. J. Lu, "Inferring Fuzzy Cognitive Map models for Gene Regulatory Networks from gene expression data," in *Bioinformatics and Biomedicine (BIBM), 2012 IEEE International Conference on*, 2012, pp. 1-4.

[54] Z. Shen and D. Huang, "A curious learning model with ELM for fuzzy cognitive maps," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* vol. 21, pp. 63-74, 2013.

[55] E. Yesil, C. Ozturk, M. F. Dodurka, and A. Sakalli, "Fuzzy cognitive maps learning using Artificial Bee Colony optimization," in *Fuzzy Systems (FUZZ), 2013 IEEE International Conference on*, 2013, pp. 1-8.

[56] M. Gregor and P. Groumpos, "Training Fuzzy Cognitive Maps Using Gradient-Based Supervised Learning," in *AIAI*, 2013, pp. 547-556.

[57] A. Kannappan and E. I. Papageorgiou, "A new classification scheme using artificial immune systems learning for fuzzy cognitive mapping," in *Fuzzy Systems (FUZZ), 2013 IEEE International Conference on*, 2013, pp. 1-8.

[58] G. Nápoles, I. Grau, R. Bello, and R. Grau, "Two-steps learning of Fuzzy Cognitive Maps for prediction and knowledge discovery on the HIV-1 drug resistance," *Expert Systems with Applications,* vol. 41, pp. 821-830, Feb. 2014.

[59] W. Stach, L. Kurgan, W. Pedrycz, and M. Reformat, "Genetic learning of fuzzy cognitive maps," *Fuzzy Sets and Systems,* vol. 153, pp. 371-401, 8/1/ 2005.

[60] T. Blickle and L. Thiele, "A Comparison of Selection Schemes Used in Evolutionary Algorithms," *Evolutionary Computation,* vol. 4, pp. 361-394, 1996/12/01 1996.

[61] A. K. Tsadiras, "Comparing the inference capabilities of binary, trivalent and sigmoid fuzzy cognitive maps," *Information Sciences,* vol. 178, pp. 3880-3894, 10/15/ 2008.

[62] S. Bueno and J. L. Salmeron, "Benchmarking main activation functions in fuzzy cognitive maps," *Expert Systems with Applications,* vol. 36, pp. 5221-5229, 4// 2009.

[63] W. Shou, W. Fan, and B. Liu, "A Data-Based Fuzzy Cognitive Map Mining Method Using DE-SQP Algorithm," in *AsiaSim*, 2012, pp. 37-43.

[64] Y. Chen, L. J. Mazlack, and L. J. Lu, "Fuzzy Cognitive Maps development using Ant Colony Optimization with local search procedure," in *Fuzzy Information Processing Society (NAFIPS), 2012 Annual Meeting of the North American*, Berkeley, California, USA, 2012, pp. 1-6.

[65] W. Stach, L. A. Kurgan, and W. Pedrycz, "Numerical and Linguistic Prediction of Time Series With the Use of Fuzzy Cognitive Maps," *Fuzzy Systems, IEEE Transactions on,* vol. 16, pp. 61-72, 2008.

[66] P. Oikonomou and E. Papageorgiou, "Particle Swarm Optimization Approach for Fuzzy Cognitive Maps Applied to Autism Classification," in *AIAI*, 2013, pp. 516-526.

[67] G. Nápoles, I. Grau, M. León, and R. Grau, "Modelling, Aggregation and Simulation of a Dynamic Biological System through Fuzzy Cognitive Maps," in *MICAI*, 2013, pp. 188-199.

[68] E. I. Papageorgiou and A. Kannappan, "Fuzzy cognitive map ensemble learning paradigm to solve classification problems: Application to autism identification," *Applied Soft Computing,* vol. 12, pp. 3798-3809, 12// 2012.

[69] E. I. Papageorgiou and W. Froelich, "Multi-step prediction of pulmonary infection with the use of evolutionary fuzzy cognitive maps," *Neurocomputing,* vol. 92, pp. 28-35, Sep. 2012.

[70] M. Crepinsek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey," *ACM Comput. Surv.,* vol. 45, pp. 1-33, 2013.

[71] M. Ronen, R. Rosenberg, B. I. Shraiman, and U. Alon, "Assigning numbers to the arrows: Parameterizing a gene regulation network by using accurate expression kinetics," *Proceedings of the National Academy of Sciences,* vol. 99, pp. 10555-10560, August 6, 2002 2002.

[72] F. Herrera, M. Lozano, and J. L. Verdegay, "Tackling Real-Coded Genetic Algorithms: Operators and Tools for Behavioural Analysis," *Artificial Intelligence Review,* vol. 12, pp. 265-319, 1998/08/01 1998.

[73] R. D. Leclerc, "Survival of the sparsest: robust gene networks are parsimonious," *Mol Syst Biol,* vol. 4, pp. 1-6, Aug. 2008.

[74] R. J. Prill, D. Marbach, J. Saez-Rodriguez, P. K. Sorger, L. G. Alexopoulos, X. Xue*, et al.*, "Towards a Rigorous Assessment of Systems Biology Models: The DREAM3 Challenges," *PLoS ONE,* vol. 5, p. e9202, 2010.

[75] D. Marbach, R. J. Prill, T. Schaffter, C. Mattiussi, D. Floreano, and G. Stolovitzky, "Revealing strengths and weaknesses of methods for gene network inference," *Proceedings of the National Academy of Sciences,* vol. 107, pp. 6286-6291, April 6, 2010 2010.

[76] R. J. Prill, J. Saez-Rodriguez, L. G. Alexopoulos, P. K. Sorger, and G. Stolovitzky, "Crowdsourcing Network Inference: The DREAM Predictive Signaling Network Challenge," *Sci. Signal.,* vol. 4, p. mr7, Sep. 2011.

[77] T. Schaffter, D. Marbach, and D. Floreano, "GeneNetWeaver: in silico benchmark generation and performance profiling of network inference methods," *Bioinformatics,* vol. 27, pp. 2263-2270, August 15, 2011 2011.

[78] G. Stolovitzky, R. J. Prill, and A. Califano, "Lessons from the DREAM2 Challenges," *Annals of the New York Academy of Sciences,* vol. 1158, pp. 159-195, 2009.

[79] C. A. Penfold and D. L. Wild, "How to infer gene networks from expression profiles, revisited," *Interface Focus,* vol. 1, pp. 857-870, December 6, 2011 2011.

[80] C. Warnecke, W. Griethe, A. Weidemann, J. S. Jurgensen, C. Willam, S. Bachmann*, et al.*, "Activation of the hypoxia-inducible factor-pathway and stimulation of angiogenesis by application of prolyl hydroxylase inhibitors," *The FASEB Journal,* vol. 17, pp. 1186-1188, June 1, 2003 2003.

[81] X. Zhang, X.-M. Zhao, K. He, L. Lu, Y. Cao, J. Liu*, et al.*, "Inferring gene regulatory networks from gene expression data by path consistency algorithm based on conditional mutual information," *Bioinformatics,* vol. 28, pp. 98-104, January 1, 2012 2012.

[82] S. Yuan, N. Tian, Y. Chen, H. Liu, and Z. Liu, "Nonlinear Geophysical Inversion Based on ACO with Hybrid Techniques," in *Natural Computation, 2008. ICNC '08. Fourth International Conference on*, 2008, pp. 530-534.