Hindawi Publishing Corporation
Advances in Bioinformatics
Volume 2016, Article ID 5283937, 9 pages
http://dx.doi.org/10.1155/2016/5283937

Research Article

Large-Scale Recurrent Neural Network Based Modelling of Gene Regulatory Network Using Cuckoo Search-Flower Pollination Algorithm

Sudip Mandal,[1] Abhinandan Khan,[2] Goutam Saha,[3] and Rajat K. Pal[2]

[1]*Department of Electronics and Communication Engineering, Global Institute of Management and Technology, Krishna Nagar, West Bengal 741 102, India*
[2]*Department of Computer Science and Engineering, University of Calcutta, Kolkata 700 098, India*
[3]*Department of Information Technology, North-Eastern Hill University, Shillong 793 022, India*

Correspondence should be addressed to Sudip Mandal; sudip.mandal007@gmail.com

Received 2 October 2015; Revised 7 January 2016; Accepted 10 January 2016

Academic Editor: Huixiao Hong

The accurate prediction of genetic networks using computational tools is one of the greatest challenges in the postgenomic era. Recurrent Neural Network is one of the most popular but simple approaches to model the network dynamics from time-series microarray data. To date, it has been successfully applied to computationally derive small-scale artificial and real-world genetic networks with high accuracy. However, they underperformed for large-scale genetic networks. Here, a new methodology has been proposed where a hybrid Cuckoo Search-Flower Pollination Algorithm has been implemented with Recurrent Neural Network. Cuckoo Search is used to search the best combination of regulators. Moreover, Flower Pollination Algorithm is applied to optimize the model parameters of the Recurrent Neural Network formalism. Initially, the proposed method is tested on a benchmark large-scale artificial network for both noiseless and noisy data. The results obtained show that the proposed methodology is capable of increasing the inference of correct regulations and decreasing false regulations to a high degree. Secondly, the proposed methodology has been validated against the real-world dataset of the DNA SOS repair network of *Escherichia coli*. However, the proposed method sacrifices computational time complexity in both cases due to the hybrid optimization process.

1. Introduction

A gene regulatory network (GRN) represents the regulatory behaviour or dependencies among a group of genes inside a cell. A GRN is characterized by a directed graph in which nodes denote genes, and regulatory dependencies among genes are depicted by directed edges between the corresponding nodes. There are two types of interaction, namely, activation and inhibition. This kind of network is unique for particular functions within a cell. Thus, the study of GRN is essential to ascertain the genetic causes of a particular disease. As a consequence of this, scientists can venture into the development of new and improved techniques for the treatment of a disease [1].

Nowadays, DNA microarrays [2, 3] are extensively utilized for the investigation of finding reasons for different illnesses. A microarray dataset contains the gene expression levels of millions of genes of a species under investigation, at a particular condition or set of conditions. Time series microarray database consists of changes in the expression of genes with time in response to some disease-causing events or any form of treatment at different time points. This includes essential information regarding the dynamic behaviour as well as the dependencies of genes.

In this work, Recurrent Neural Network (RNN) [4] which is a closed loop Neural Network with a delayed feedback has been used to model dynamics and dependencies of the genetic system from temporal genetic data. Using suitable

optimization methods, RNN based GRN can be inferred for the purpose, where the objective function of optimization is chosen so that it becomes proportional to the training error. Different metaheuristics techniques, namely, Genetic Algorithm (GA) [5], Particle Swarm Optimization (PSO) [6], K-Means Population-Based Incremental Learning (KPBIL) [7], Invasive Weed and Artificial Bee Colony (ABC) [8], PSO and Ant Colony Optimization (ACO) [9], Bat Algorithm [10], have been implemented to infer GRNs from the time series microarray data. Computational reconstruction of GRN is essentially a reverse engineering problem. All the approaches mentioned above are tested against both small-scale artificial and real-life GRNs. However, Noman et al. [11] proposed Decoupled Recurrent Neural Network, which was trained by Differential Evolution (DE), where a penalty term or L1 regularizer was introduced into the objective function to balance between the accuracy of the parameters and the actual network structure. The model mentioned above [11] is very efficient in finding all the valid regulations and accurately reproducing the dynamics of both of the small networks studied. However, the main disadvantage of this model is that the given model predicts a large number of false regulations for more extensive or larger networks. The balance between fitness value and actual network structure for large-scale networks is the primary concern of the reverse engineering problem of GRN, and it is still an open area of research.

In this paper, a hybrid Cuckoo Search (CS)-Flower Pollination Algorithm (FPA), CS-FPA, is proposed for the inference of GRNs from time-series data. FPA is used to train the RNN parameters, and CS is introduced to find the biologically plausible network architecture to select the best combination of genes that are responsible for modifying the expression of each gene. The preliminary notions of RNN, CS, and FPA are discussed in the next section. The details of the fitness function of FPA for decoupled RNN and the learning process for finding the actual structure of a GRN using CS are discussed in Section 3. Next, the effectiveness of the proposed CS-FPA based RNN model is tested against a large artificial GRN without the presence of noise as well as with noisy data. The result is also compared with other state-of-the-art methods. The conclusion is given in Section 4.

2. Theoretical Background

We have discussed the fundamental theoretical concepts of RNN, CS, and FPA, in this section, for a better understanding of the proposed methodology.

2.1. Preliminaries of RNN. The RNN model [12] is a closed loop Artificial Neural Network that has a delay variable, between the outputs of each neuron in the output layer of the RNN, to each of the neurons in the input layer, which is suitable to model temporal behaviour or dynamics of data (Figure 1). For a canonical RNN model [4–9, 11] it is assumed that each of the total N output neurons in the unit is a gene expression value of next time instant $e_i(t + \Delta t)$, and the input neurons are the gene expression of present state $e_i(t)$ for

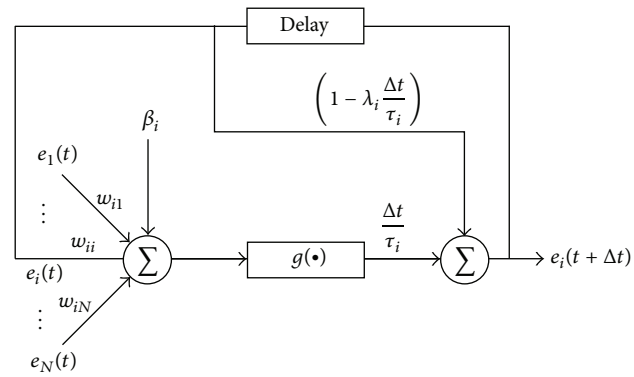

Figure 1: A neuron in the RNN model [7].

the same genes; thus they interact with each and every one in regenerative way:

$$e_i(t + \Delta t) = \frac{\Delta t}{\tau_i} f\left(\sum_{j=1}^{N} w_{i,j} e_j(t) + \beta_i\right) + \left(1 - \frac{\Delta t}{\tau_i}\right) e_i(t) \quad \text{where } i = 1, 3, \ldots, N. \tag{1}$$

Here, $f(\cdot)$ is usually a sigmoid function; $f(z) = 1/(1+e^{-z})$ is used as a classification function; and $w_{i,j}$ is the weight of inputs of the RNN model, and it stands for the type and strength of the regulatory interaction of the jth gene with the ith gene. From the point of view of a GRN, each node corresponds to a gene, and a connection between two nodes defines their interaction. The weight values can be either positive, negative, or zero; $w_{i,j}$ is the most significant term for a GRN as the value of $w_{i,j}$ is the connecting weight of an edge of the GRN, which represent the connections between gene-i and gene-j. A positive value of $w_{i,j}$ represents activation of gene-i by gene-j, a negative value denotes repression or inhibition of gene-i by gene-j, and $w_{i,j} = 0$ means that gene-j has no regulatory control on gene-i. The term β_i represents the basal expression level or a bias term, and τ_i is a time constant (delay) of the ith gene; Δt is incremental time instance; in this work it is always set as 1. Thus, any RNN model can be expressed by a set of $N(N + 2)$ parameters, $\Omega = \{w_{i,j}, \beta_i, \tau_i\}$, where $i, j = 1, 2, \ldots, N$.

2.2. Preliminaries of Cuckoo Search (CS) Optimization. Yang [13] first proposed Cuckoo Search optimization [14–18] based on brood parasitism of cuckoo birds that reproduce their eggs by utilizing nests of other host birds. These birds have the ability to use other birds for raising the new generation. Cuckoo lay their eggs, one or more than one, in the nest of the host birds in their absence using Lévy flight. Lévy flight [19] is an important characteristic of CS. Levy flight is defined as a random movement done by the birds with a step value of distributed probability. While new solution x^{t+1} for a cuckoo i, Lévy flight is performed as

$$x_i^{t+1} = x_i^t + \alpha \oplus \text{Lévy}(\lambda). \tag{2}$$

Table 1: RNN model parameters of large artificial network [11].

$w_{i,j}$	$w_{1,14} = -15, w_{5,1} = 10, w_{6,1} = -20, w_{7,2} = 15, w_{7,3} = 10, w_{8,4} = 20, w_{9,5} = -20, w_{9,6} = 10, w_{9,17} = 10, w_{10,7} = -10,$ $w_{11,4} = -15, w_{11,7} = 15, w_{11,22} = -15, w_{12,23} = 10, w_{13,8} = 20, w_{14,9} = 15, w_{15,10} = -10, w_{16,11} = 15, w_{16,12} = -15,$ $w_{17,13} = -20, w_{19,14} = -15, w_{20,15} = 10, w_{21,16} = -20, w_{23,17} = -10, w_{24,15} = -15, w_{24,18} = -20, w_{24,19} = 15,$ $w_{25,20} = -10, w_{26,11} = 20, w_{26,28} = 20, w_{27,24} = -15, w_{27,25} = 10, w_{27,30} = 15, w_{28,25} = -15, w_{29,26} = 10, w_{30,27} = 15,$ others $w_{i,j} = 0.0$
β_i	$\beta_i = 5$ for $i = \{2, 5, 6, 10, 16, 24, 28\}$, $\beta_i = -5$ for $i = \{15, 17, 27\}$ otherwise $\beta_i = 0$
τ_i	$\tau_i = 10$ for $i = \{1, 2, \ldots, 30\}$

Here $\alpha > 0$ is the step size; $\oplus$ means entrywise multiplications. Levy flights essentially provide a random walk and the random steps are drawn from a Lévy distribution [13, 19] as follows:

$$\text{Lévy}(\lambda) \sim \frac{\lambda \Gamma(\lambda) \sin(\pi\lambda/2)}{\pi} \frac{1}{s^{1+\lambda}}. \quad (3)$$

In this equation, $\Gamma(\lambda)$ is the standard gamma function. This distribution is valid for large steps $s > 0$. For optimization problems, one cuckoo nest corresponds to one solution of an optimization problem. When the host bird recognizes the alien eggs, the host bird may destroy the eggs or may leave its nest and build a new nest with a certain probability p_a. To avoid this, cuckoos learn to make eggs similar to the host bird's eggs. However, the highest quality nest with eggs (i.e., best solutions) will be selected to move over to the next generation where the quality of an egg or fitness of a solution is simply proportional to the value of objective function.

2.3. Preliminaries of Flower Pollination Algorithm (FPA). FPA [20] is typically associated with the transfer of pollen for reproduction or flowering of plants, and pollinators such as insects, birds, and bats are mainly responsible for such transfer. FPA [21–23] is a recently proposed metaheuristic that is based on some simplified rules for pollination. Biotic cross-pollination can be assumed as a process of global pollination, and pollen carrying pollinators follow Lévy flights during transport (Rule 1). For local pollination, abiotic pollination and self-pollination are used (Rule 2). Pollinators may develop flower reliability, which is proportional to the resemblance of two flowers, that is, reproduction probability (Rule 3). The switching of local to global pollination can be controlled by a switch probability $p \in [0, 1]$, slightly biased towards local pollination (Rule 4). Here, each pollen or flower corresponds to a solution of the optimization problem being considered.

Global and local pollination (i.e., search) are done according to the following two equations [21], respectively:

$$x_i^{t+1} = x_i^t + \gamma\, \text{Lévy}(\lambda) \left(g_* - x_i^t\right), \quad (4)$$

$$x_i^{t+1} = x_i^t + \varepsilon \left(x_j^t - x_k^t\right). \quad (5)$$

Here, x_i^t is the pollen i or solution vector x_i at iteration t, γ is the scaling factor to control the step, g_* is the current best solution found among all solutions at the current iteration, x_j^t and x_k^t are pollens from the different flowers of the same plant species, and ε stands for random walk step size within a uniform distribution in $[0, 1]$. The reason behind selecting FPA as optimization method is that it gives better convergence and accuracy than other popular metaheuristic techniques [21].

3. Methodology

The RNN formalism is based on a set of parameters, $w_{i,j}$, β_i, τ_i, which we refer to as RNN model parameters in this work. The reverse engineering of GRNs from temporal microarray data requires finding the optimum values of the RNN parameters with the help of optimization techniques such that the training error is minimized.

3.1. Model Used for Validation. We choose a large artificial network with 30 genes, to investigate the inference capability of the proposed algorithm. This network structure is very sparse in nature, and it has already been studied in [11]. The parameters of this architecture were chosen arbitrarily as shown in Table 1 where there are only 36 connections or regulations in the network; for example, $w_{1,14} = -15$, signifying an edge (a regulatory relationship) that exists between Gene 1 and Gene 14 in the GRN, and the negative value implies that Gene 14 inhibits or suppresses the expression of Gene 1. Any negative value denotes inhibition or repression while a positive value implies activation. The numerical value has only mathematical significance for the RNN formalism implemented here. Only the nature of the value of $w_{i,j}$, that is, whether $w_{i,j} < 0$, 0, or > 0, has biological significance.

The parameters β_i and τ_i also have no significance in the biological structure of a GRN. They are required for the mathematical modelling of the RNN formalism. They basically determine the bias and rate of change of dynamics for a particular gene. Using (1), 5 time series datasets, with 50 time points in each, are generated for learning of RNN assuming that $\Delta t = 1$. These time series data are used as the training data, represented by a gene expression matrix where each column indicates a gene and each row indicates a time point; the data in each cell means the gene expression level of a particular gene at a particular time point.

The corresponding GRN is shown in Figure 2, where arrow-head and T-head denote activation and suppression, respectively, for positive and negative weights, respectively, between any two genes.

3.2. Decoupled Recurrent Neural Network Training. In all optimization methods, an objective/fitness function is used to measure the quality of a solution. One of the most regularly

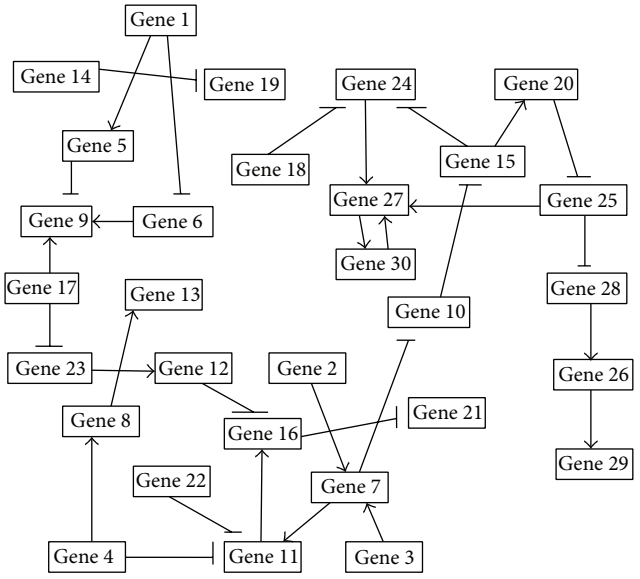

Figure 2: GRN using RNN parameters as described in Table 1.

implemented estimation criteria is the squared error that is defined as follows:

$$fob = \sum_{k=1}^{M} \sum_{i=1}^{N} \sum_{t=1}^{T} \left(e_{\text{cal},k,i,t} - e_{\text{exp},k,i,t} \right)^2. \quad (6)$$

In the above equation, N is the number of genes in the GRN, T is the number of sampling instances of the observed gene expression data, and M is the number of training datasets; $e_{\text{cal},k,i,t}$ is the numerically calculated gene expression value of the ith gene at time t, for the kth dataset; $e_{\text{exp},k,i,t}$ is the actual gene expression level of the ith gene at time t, for the kth dataset. The function fob represents the total squared error between the calculated and experimental gene expression data.

Thus, the training of the RNN model parameters is, in essence, a nonlinear optimization problem. The primary objective is to determine the optimal RNN parameters such that the mean square error is minimised, and the calculated gene expression data fits with the observed gene expression data. Since, for N genes, $N(N + 2)$ parameters must be determined to find the solution of a set of equations as in (1), the RNN model parameter search space is of $N(N + 2)$-dimensional space (here 960). However, this space becomes too computationally expensive in the case of large-scale genetic networks. Therefore, the problem of inferring a GRN from temporal expression data is decoupled or divided into several subproblems corresponding to each of the genes. Now, the objective of each subproblem corresponding to ith gene is to find the values of decoupled RNN model parameters which minimizes the prediction. Thus, we define the objective function for the ith gene only as

$$fob_i = \sum_{k=1}^{M} \sum_{t=1}^{T} \left(e_{\text{cal},k,i,t} - e_{\text{exp},k,i,t} \right)^2. \quad (7)$$

Hence, to solve the differential equation (1), the number of RNN parameters needed to determine is only $(N + 2)$ parameters for the ith gene only. Thus, this decoupling method divides a $N(N + 2)$-dimensional problem N subproblems of dimension $(N + 2)$. Accumulating the $(N + 2)$ parameters of all N genes, the overall structure of the RNN model can be achieved which in turn will define the inferred GRN.

3.3. CS and FPA Hybridization. Real-life GRNs are sparsely connected; that is, very few connections exist among genes, and the measured data are also very noisy. Thus, for GRN inference problems, it is expected that the values of a majority of the weight parameters are zero. Nevertheless, it is found that though the complexity is reduced up to a certain limit by using a decoupled strategy, it may lead to different solutions. Each solution may have very low error value depending on the different connectivities or structures among the genes in the GRN and the corresponding values of the kinetic parameters. This is known as the overfitting problem that occurs mainly due to the nonlinearity of the RNN methodology, large search space, and a large number of parameters (here 32 for the decoupled RNN formalism, i.e., still quite large) to be optimized. Thus, to overcome this problem in real-life genetic networks, a balance between prediction error minimization and actual regulatory structure of GRN needs to be achieved.

The proposed Cuckoo Search-Flower Pollination Algorithm hybrid approach utilizes a single-layer RNN formalism to determine the weights of the edges between the input and output layers of RNN, with the help of FPA that minimizes the error between the calculated and experimental gene expression profiles. This RNN structure is hybridized with a new metaheuristic CS algorithm for creating hypothetical interconnections or regulatory edges among genes by selecting the best combination of affecting/regulatory genes. Moreover, the maximum connectivity of each gene is restricted to $I_{\max}$ as it is observed that real-life GRNs are sparsely connected; that is, very few genes participate in actual regulations.

In our present problem, we have chosen the value of $I_{\max} = 3$; that is, we assume that at most any three genes can affect any single target gene. CS technique has been introduced to detect these three regulatory genes, and FPA is used to know the weight of the RNN model where these three regulatory genes act as the input node of RNN and the target gene acts as the output node. The RNN is implemented to determine how the gene expression levels of those three genes, at a particular time point, affect the target gene's expression value at the next time instance. Assuming this, we assign the value of $w_{i,j}$ to be zero except for those three regulatory genes and the search space thus becomes 5-dimensional (3 weight, 1 bias, and 1 delay parameter). It may be possible that, for a particular regulatory gene set, one or more $w_{i,j}$ becomes zero after optimization using FPA. This implies no regulation for those genes with zero value of $w_{i,j}$.

The CS is initialized randomly with a population of different solutions or eggs on the different random nest, and the quality of each egg in the host nest is calculated using FPA based RNN to observe the impact of regulatory genes on other genes in the GRN as part of one generation. However, we initialized with a population of 10,

each consisting of 3 different combinations of genes starting from $(1, 2, 3), (4, 5, 6), \ldots, (28, 29, 30)$. This type of predefined initialization is used as it increases the probability of covering all regulatory genes without repetition. Search range for the model is chosen as previous work [11] like $w_{i,j} = [-25, 25]$, $\beta_i = [-10, 10]$, and $\tau_i = [0, 15]$, respectively.

The quality of a host nest or fitness of a solution is simply proportional to the objective function that is the resultant error of RNN for the set of particular genes or nest. FPA always tries to minimize the training error by optimizing the value of the RNN model parameters. During training of RNN, each pair in the training dataset contains the gene expression values for only 3 regulatory genes from the one time instance of the microarray data to be the input values to the RNN model, and the expression value of the target gene at the next time instance of the microarray data is the target output of the RNN. This helps to reduce the execution time by reducing the unnecessary calculations corresponding to nonregulatory genes. Moreover, it is observed that during optimization if the fitness value for FPA based RNN becomes less than 1×10^{-8}; then it gives the almost actual value of the parameters of RNN for the regulatory genes. Therefore, a stopping criterion is introduced to minimize the execution time of the algorithm; that is, if the fitness value for a particular gene set becomes less than 1×10^{-8} after some iterations, the program stops execution instantly. Moreover, the corresponding genes set becomes the desired output.

Sometimes, it is found that if any solution does not consist of the actual regulatory genes, then the convergence is very slow which may increase the execution time as it is not able to go below the value 1×10^{-8} even after a large number of iterations. Therefore, another alternative stopping criterion is also imposed which can be stated as follows: if the difference between current best fitness value and fitness value of the (current-200)th iteration is less than 1×10^{-10}, then the program execution will also stop. Better host nest with better quality eggs of each generation will move to the next generations. After successful completion of all iterations, we have a set of regulatory genes which can affect the target gene most. This process is repeated for all 30 genes one by one to get the final GRN structure.

Here inputs are the time series data, generated by (1), using parameters as shown in Table 1, and the outputs are a combination of 3 regulatory genes (jth) for a particular target gene (ith) along with the value of regulatory weights, that is, $w_{i,j}$, corresponding to those 3 regulatory genes for which the minimum or optimum fitness value is achieved. It is worth mentioning that for a target gene we always get a combination of three responsible genes, but this does not mean that during GRN reconstruction there will always be regulatory edges from those regulatory genes towards target gene. The existence of a directed edge in a network also depends on the weights between target and regulatory genes. If amplitudes of weights are zero or very small, there will be no edges for the target gene; that is, there will be noninteraction. Small perturbation of obtained weights from actual one can be ignored unless and until it does not change the polarity or sign, that is, the type of regulation. Moreover, small values of weights can be considered as zero. It is believed that if the number of iterations and population size are large enough, these small perturbations can also be avoided, but execution time will also increase consequently. Thus, during reconstruction of GRN, both the set of regulatory genes and the value of the corresponding gene's weights must be kept in mind. The pseudocode of this CS-FPA hybrid is given as in Algorithm 1.

3.4. Results for Artificial Dataset. After execution of the proposed methodology, we get the hypothetical interconnections among genes regarding weights of the RNN model. Now, the performance of any inference algorithm for GRN is measured regarding sensitivity (S_n), specificity (S_p), accuracy, and Matthews Correlation Coefficient (MCC) [11, 24] which are defined as follows:

$$
\begin{aligned}
S_n &= \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \\
S_p &= \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}, \\
\text{Accuracy} &= \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}, \\
\text{MCC} &= \frac{(\mathrm{TP} \times \mathrm{TN}) - (\mathrm{FP} \times \mathrm{FN})}{\sqrt{(\mathrm{TP} + \mathrm{FP}) \times (\mathrm{TP} + \mathrm{FN}) \times (\mathrm{TN} + \mathrm{FP}) \times (\mathrm{TN} + \mathrm{FN})}}.
\end{aligned}
\tag{8}
$$

TP (True Positive) denotes the number of correctly predicted regulations, and TN (True Negative) represents the number of properly predicted nonregulations. FP (False Positive) denotes the number of incorrectly predicted regulations, and FN (False Negative) represents the number of falsely predicted nonregulations by the inference algorithm. All the experiments have been performed using MATLAB R2009b, running on Windows 7 with an Intel© Dual Core processor and 2 GB of RAM. In this work, both noiseless and noisy data are considered for the construction of the artificial network to validate the proposed algorithm. Validation of the proposed method is achieved in terms of correct prediction of RNN model parameters, that is, correct prediction of regulations in the GRN. The artificial time series datasets generated using (1) are used as training inputs.

For the reconstruction of a GRN consisting of 30 genes, we need an adjacency matrix (here weight matrix of RNN) of 30×30 dimension. In this case, there are only 36 regulations, that is, 36 nonzero value of weights (see the Supplementary Material available online at http://dx.doi.org/10.1155/2016/5283937). For noiseless data, using RNN model, we found that the proposed algorithm can predict 32 regulations correctly compared to those which were present in actual GRN, and also includes only four unwanted regulations. Thus, TP = 32, FP = 4, FN = 4, and TN = 860. It is found that the proposed algorithm is able to detect maximum correct regulations or TPs with a very good accuracy (for maximum cases, the fitness value went below 10^{-8}). Though this method only detects 4 false regulations, 4 true regulations are also missing. Nevertheless its inference capability or performance is quite

For gn (gene) = 1 : 30
  Initialize the population *para* with $n = 10$ host nests (solutions) with
  dimension $I_{\max} = 3$ that is $(1, 2, 3); (4, 5, 6); \ldots, (28, 29, 30)$.
  For $n = 1 : 10$
  Calculate *fob* (fitness value) for all 10 solutions using RNN and FPA;
  If $(fob_n < 1 * 10^{-8})$
    Break;
  End if;
  End for;
 If (minimum $(fob) > 1 * 10^{-8}$)
  For $j = 1$: max iteration (100)
    Randomly select a cuckoo (kth) avoid current best;
    Randomly generate another nest (sth) keeping current best nests by Lévy flights;
    If $(fob_s < fob_k)$
    k is replaced by the news solution;
    End if;
    If $(fob_k < 1 \times 10^{-8})$
    Break;
    End;
    Discard the worse nests with a fractional probability ($p_a = 0.25$);
      Keep the highest quality nest that is best solution with best fitness value;
      Rank the available solutions and locate the current best;
  End for;
 End if;
End for;
Post-processing and visualization of GRN;
Function *fob* (*para, gn*)
  Initialize a *rnnpara* population of *nf* (30) with dimension *df* (5) pollen randomly with a switch probability $p_f = 0.8$
  Find the fitness *fun* for all solutions and best pollen among them
  For (*tf* < MaxGeneration (2000))
    For $i = 1 : nf$
      If rand $< p_f$,
        Draw a (df-dimensional) step vector L which obeys a Lévy distribution
        Global pollination via equation (4)
      Else
        Draw ϵ from a uniform distribution in $[0, 1]$
        Randomly choose *jf* and *kf* among all the solutions
        Do local pollination via equation (5)
      End if
        Evaluate fitness *fun* of new solutions of pollens
      If new solutions are better
        Update them in the population
      End if
    End for
    Find store the best fitness of current iteration
    If ((*bestfitness* (*tf*) $< 1 * 10^{-8}$) ‖ ((*tf* > 500) && ((*bestfitness* (*tf* − 200) − *bestfitness* (*tf*)) $< 1 * 10^{-10}$)))
      Break
    End if
  End for
  Return bestfitness
End *fob*
Function *fun* (*rnnpara*, *para*, *gn*)
  Define M (5) times series data with *T (50)* sample point
  Calculate the gene expression value of next time instance using equation (1)
  Determine the squared error using equation (7)
  Return error
End *fun*

Algorithm 1

Table 2: A comparative study of the performance of the large-scale artificial network.

Data type	Method	TP	TN	FP	FN	S_n	S_p	Accuracy	MCC
Noiseless	CS-FPA	32	860	4	4	0.889	0.995	0.991	0.884
	DE [11]	22	861	3	14	0.611	0.996	0.981	0.725
5% noise	CS-FPA	32	845	19	4	0.889	0.978	0.974	0.735
	DE [11]	11	848	16	25	0.305	0.981	0.954	0.329

satisfactory and better than other state-of-the-art methods like DE [11].

Next, we add 5% random noise to the initial training dataset and apply the CS-FPA hybrid algorithm to it to check the robustness against noise like real-world data. It is observed that, in the presence of noise, the number of FPs increased significantly. However, it can still identify all the TPs (previously inferred) with good accuracy. For noisy data, TP = 32, FP = 19, FN = 4, and TN = 849. Table 2 shows a comparative study of the performance of the proposed methodology with earlier work with respect to specificity and sensitivity.

It is quite interesting to observe that performance (in terms of sensitivity) of CS-FPA does not change for detecting true positive regulations in the presence of noise whereas sensitivity of DE [11] drastically falls to very low value in the presence on noise that denotes inability of DE [11] for finding actual regulation in the presence of noise. Moreover, specificity, accuracy, and MCC are better than DE [11], and values of these parameters do not change significantly due to noise though some false regulations are included into the GRN. However, due to a parallel implementation of two metaheuristic for one problem, overall computational time complexity is still significant although several break conditions are applied to minimize the execution time as much as possible which is 1.5 hours on average for this large network model.

3.5. Results for Real-Time Dataset of E. coli. Microarray experiments on the SOS DNA repair network for *E. coli* [25] were first done by the Uri Alon group [26]. The experimental datasets are considered as the benchmark for the evaluation of algorithms for reverse engineering of GRNs from real-world datasets. In the SOS network, 8 genes were considered (*uvrD, lexA, umuD, recA, uvrA, uvrY, ruvA,* and *polB* as shown in Figure 3) due to their significant involvement in the process of DNA repair. During their experiments, the *E. coli* cells were irradiated with UV light. Four experiments were performed with different UV light intensities. Each experiment consists of 50 time steps spaced by 6 minutes for each of the eight genes. However, in this work, the first dataset with all eight genes has been considered where it is being preprocessed by neglecting first time point (zero) normalizing in the range $[0, 1]$.

Search range for the model is chosen as previous work [11] like $w_{i,j} = [-10, 10]$, $\beta_i = [-10, 10]$, and $\tau_i = [0, 10]$, respectively. For CS, the value of N is set as 2 as the number of available genes is limited to 8 only and other parameter settings remain the same. If we apply CS-FPA hybrid to the SOS dataset, only 4 out of 9 potential regulations can be appropriately predicted from the data, which are the inhibition of *lexA* on *uvrD*, *umuD, recA,* and *ruvA*. Moreover, it also includes 11 false regulations in the network which is a disadvantage of this process. The results are shown in Table 3.

Table 3: Results obtained for the *E. coli* SOS DNA repair network.

	uvrD	*lexA*	*umuDC*	*recA*	*uvrA*	*uvrY*	*ruvA*	*polB*
uvrD	0	−	0	0	0	0	0	0
lexA	+	0	0	0	0	+	0	0
umuDC	−	−	0	0	0	0	0	0
recA	0	−	0	0	−	0	0	0
uvrA	0	0	+	0	0	0	−	0
uvrY	0	0	0	−	0	0	0	+
ruvA	0	−	0	0	0	0	0	+
polB	0	0	−	−	0	0	0	0

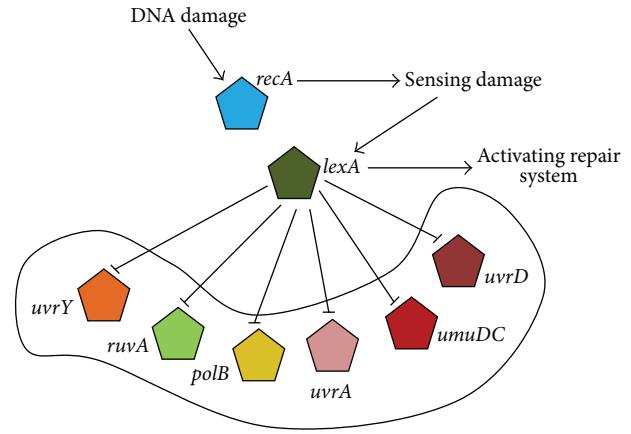

Figure 3: The graphical representation of the actual SOS network for *E. coli* [7].

Furthermore, using two or more time series does not yield any enhancement for the results. The cause behind this is that inference of real-time GRNs is an ill-posed problem that has no unique solution. Noise and measurement error in this type of real-time series microarray is another issue. This inherent difficulty is a limitation of our proposed method.

4. Conclusion

Various researchers have already proposed numerous techniques to solve the reverse engineering problem of GRNs from temporal genetic expression data in the domain of computational biology and bioinformatics. It is imperative to enhance the accuracy of the inference algorithms as well as to reduce the number of incorrect predictions (i.e., FPs) within a plausible runtime.

The RNN formalism is a very popular candidate for inferring GRNs from microarray gene expression data regarding biological plausibility and computational efficiency. In this work, we have implemented the decoupled RNN model where the regulatory parameters of each gene are calculated independently in separate search instances. We incorporated hybridized technique where two metaheuristics are paired to obtain the RNN based GRN model with less search space, less computational complexity, and more accuracy. In this paper, hybridized CS-FPA is proposed for reconstruction of GRNs where FPA is used to train the RNN parameters, and CS is introduced to select the best combination of genes that are responsible for modifying the expression of each gene. Moreover, the maximum connectivity of each gene is restricted to $I_{\max}$ as it is observed that real-life GRNs are sparsely connected; that is, very few genes participate in regulations.

To prove the efficiency of this inference algorithm, it is applied to a benchmark problem of the artificial network with 30 genes with and without noise. With the use of fewer data points, CS-FPA based RNN can infer the network with very high accuracy. However, in the presence of noise, the number of FPs increases significantly, but it can still identify all TPs (inferred in the noiseless scenario) with good accuracy. It is also found that noise robustness is better than other existing methods for artificial data. In the instance of the *E. coli* dataset, it can detect only four true regulations and includes some false regulations.

Another important observation that is apparent from our results is that the proposed methodology can reconstruct the large artificial GRNs more efficiently than that of real-life GRNs. However, this needs further study on different networks available to us, and the existing boundary of our work validates this observation. In the future, various regularization techniques, the inclusion of prior knowledge about GRNs, and parallel computing methods may be utilized to improve the accuracy and speed further.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

References

[1] N. Fankhauser, I. Cima, P. Wild, and W. Krek, "Identification of a gene expression signature common to distinct cancer pathways," *Cancer Informatics*, vol. 11, pp. 139–146, 2012.

[2] D. R. Masys, "Linking microarray data to the literature," *Nature Genetics*, vol. 28, no. 1, pp. 9–10, 2001.

[3] J. Quackenbush, "Microarray data normalization and transformation," *Nature Genetics*, vol. 32, no. 5, pp. 496–501, 2002.

[4] J. Kolen and S. Kremer, *A Field Guide to Dynamical Recurrent Networks*, IEEE Press, New York, NY, USA, 2001.

[5] J.-H. Chiang and S.-Y. Chao, "Modeling human cancer-related regulatory modules by GA-RNN hybrid algorithms," *BMC Bioinformatics*, vol. 8, article 91, 2007.

[6] R. Xu, I. I. D. Wunsch, and R. Frank, "Inference of genetic regulatory networks with recurrent neural network models using particle swarm optimization," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 4, no. 4, pp. 681–692, 2007.

[7] L. Palafox, N. Noman, and H. Iba, "Study on the use of evolutionary techniques for inference in gene regulatory networks," in *Natural Computing and Beyond*, vol. 6 of *Proceedings in Information and Communications Technology*, pp. 82–92, Springer, Tokyo, Japan, 2013.

[8] P. Rakshit, P. Das, A. Konar, M. Nasipuri, and R. Janarthanan, "A recurrent fuzzy neural model of a gene regulatory network for knowledge extraction using invasive weed and artificial bee colony optimization algorithm," in *Proceedings of the 1st International Conference on Recent Advances in Information Technology (RAIT '12)*, pp. 385–391, Dhanbad, India, March 2012.

[9] K. Kentzoglanakis and M. Poole, "A swarm intelligence framework for reconstructing gene networks: searching for biologically plausible architectures," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 2, pp. 358–371, 2012.

[10] S. Mandal, A. Khan, G. Saha, and R. K. Pal, "Reverse engineering of gene regulatory networks based on S-systems and bat algorithm," *Journal of Bioinformatics and Computational Biology*, 2016.

[11] N. Noman, L. Palafox, and H. Iba, "Reconstruction of gene regulatory networks from gene expression data using decoupled recurrent neural network model," in *Natural Computing and Beyond: Winter School Hakodate 2011, Hakodate, Japan, March 2011 and 6th International Workshop on Natural Computing, Tokyo, Japan, March 2012, Proceedings*, vol. 6 of *Proceedings in Information and Communications Technology*, pp. 93–103, Springer, Berlin, Germany, 2013.

[12] J. Vohradský, "Neural network model of gene expression," *The FASEB Journal*, vol. 15, no. 3, pp. 846–854, 2001.

[13] X. S. Yang, *Engineering Optimization: An Introduction with Metaheuristic Applications*, John Wiley & Sons, New York, NY, USA, 2010.

[14] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the World Congress on Nature and Biologically Inspired Computing (NaBIC '09)*, pp. 210–214, Coimbatore, India, December 2009.

[15] X.-S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.

[16] P. Civicioglu and E. Besdok, "A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms," *Artificial Intelligence Review*, vol. 39, no. 4, pp. 315–346, 2013.

[17] A. S. Jereesh and V. K. Govindan, "Gene regulatory network modeling using cuckoo search and S-system," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 9, pp. 1231–1237, 2013.

[18] G.-G. Wang, A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "A new hybrid method based on krill herd and cuckoo search for global optimisation tasks," *International Journal of Bio-Inspired Computation*, vol. 10, no. 150, pp. 1–13, 2015.

[19] I. Pavlyukevich, "Lévy flights, non-local search and simulated annealing," *Journal of Computational Physics*, vol. 226, no. 2, pp. 1830–1844, 2007.

[20] Wikipedia article on pollination, http://en.wikipedia.org/wiki/Pollination.

[21] X.-S. Yang, "Flower pollination algorithm for global optimization," in *Unconventional Computation and Natural Computation*, vol. 7445 of *Lecture Notes in Computer Science*, pp. 240–249, Springer, Berlin, Germany, 2012.

[22] X.-S. Yang, M. Karamanoglu, and X. S. He, "Flower pollination algorithm: a novel approach for multiobjective optimization," *Engineering Optimization*, vol. 46, no. 9, pp. 1222–1237, 2014.

[23] X.-S. Yang, M. Karamanoglu, and X. S. He, "Multi-objective flower algorithm for optimization," *Procedia Computer Science*, vol. 18, pp. 861–868, 2013.

[24] N. Noman and H. Iba, "Reverse engineering genetic networks using evolutionary computation," *Genome Informatics*, vol. 16, no. 2, pp. 205–214, 2005.

[25] J. J. Faith, B. Hayete, J. T. Thaden et al., "Large-scale mapping and validation of *Escherichia coli* transcriptional regulation from a compendium of expression profiles," *PLoS Biology*, vol. 5, no. 1, article e8, 2007.

[26] 2016, http://www.weizmann.ac.il/mcb/UriAlon/.