Study on the Use of Evolutionary Techniques for Inference in Gene Regulatory Networks

Leon Palafox[1], Nasimul Noman[2], and Hitoshi Iba[2]

[1] Department of Electric Engineering, University of Tokyo
[2] Department of Information, Science and Technology, University of Tokyo
{leon,noman,iba}@iba.t.u-tokyo.ac.jp

Abstract. Inference in Gene Regulatory Networks remains an important problem in Molecular Biology. Many models have been proposed to model the relationships within genes in a DNA chain. Many of these models use Evolutionary Techniques to find the best parameters of specific DNA motifs.

In this work, we compare the popular S-System using a powerful evolutionary technique, DPSO, and the novel Recursive Neural Network model, using clustered Population Based Incremental Learning (PBIL). We will use the SOS network for *E.coli* to do the comparison to finally show how they fare against other techniques in the area of Gene Regulatory Network (GRN) inference.

1 Introduction

Finding the correct interactions within genes' motifs is an important problem in Molecular Biology. There are many techniques to find the different interactions between genes in a Gene Regulatory Network (GRN). Most of these techniques have different strengths and weaknesses as we try to find interactions in larger networks.

In this work, we present a comparison between two techniques used to find the best parameters of a GRN. On one side, we use the S-System, an ODE modeling framework to find the correct interactions, which has been used by many groups in the area [1,2]. On the other hand, we use the novel Recursive Neural Network (RNN) architecture, recently proposed as a surrogate to model the interactions in a GRN [3].

To find the best parameters in both the S-System and the RNN, we use two different evolutionary techniques, we compare the effectives of random jumps of the Dissipative Particle Swarm Optimization (PSO)[4] against clustering the candidate solutions to enrich the search space in Clustered Population Based Incremental Learning (PBIL), which we call K-Means PBIL (KPBIL). We use PSO to solve the S-System and KPBIL to solve the RNN.

The paper is organized as follows; first, we present a brief introduction to both techniques, we show how to use the S-System and RNN to do reverse engineering of GRN. Then, we introduce the two evolutionary techniques we used, Dissipative PSO and KPBIL. Then, we describe how to do inference for the parameters over small sets and finally use both algorithms to do inference in the real SOS network for *E.coli*.

Y. Suzuki and T. Nakagaki (Eds.): WSH 2011 and IWNC 2012, PICT 6, pp. 82–92, 2013.

1.1 Related Work

Different researchers have attacked the problem of gene network inference using different techniques. These techniques have strengths and weaknesses, of which we will describe some.

One of the most intuitive approaches is the use of graphical models to represent the genes and its connections, Akutsu [5] used a boolean network to represent the connections between genes as logical connections. This model, albeit powerful is limited in its capability of representing the true state of genes over time, and was restricted to tree-like structures. Murphy [6], however, showed that these models could be best represented as dynamic Bayesian networks, which allow for structures other than trees to represent data along time. Bayesian Networks, however, have difficulties to deal with the data when it is represented in loops, like in real GRN.

Other family of widely used models are the differential equations based models (ODE), these models use non linear differential equations to represent dynamic data. The S-System, an ODE modeling framework, is extensively used by many researchers in the area of evolutionary computation [2,7,8]. ODE modeling frameworks have many advantages compared to graphical representations, since they allow for a richer representation of the data, however, solving a set of differential equations is computationally more expensive than inferring probabilistic graphical models, making them unfeasible to work with larger networks.

Vohradský[9], proposed using RNNs to model gene networks. Furthermore, Xu et al [3] successfully applied Particle Swarm Optimization (PSO) to find the parameters of the RNN. Different works [10,11] have used evolutionary computation techniques to solve Neural Network's architectures with good results. This model has some of the advantages of the graphical models, like the scalability, and the advantage of richness of expression that the ODE modeling frameworks offer.

2 GRN Modeling

To model the GRN, we use training data in the form of microarray data, this data is the expression of different genes over a period of time. We use two different models and compare them, the S-System and the RNN model.

2.1 S-System as a Model for Biological Networks

The S-System, first proposed by Savageau [12], provides a mathematical framework to represent and analyze biological systems. It represents a network as a set of differential equations having the form:

$$\frac{dX_i}{dt} = \alpha_i \prod_{j=1}^{N} X_j^{g_{ij}} - \beta_i \prod_{j=1}^{N} X_j^{h_{ij}} \tag{1}$$

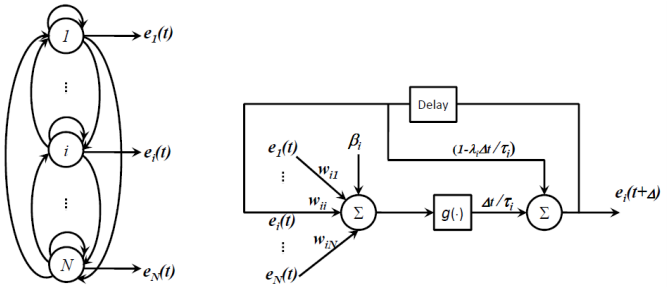


Fig. 1. Left: RNN's graphical model with the observed data, Right:RNN scheme wit the addition of delay terms and the output for a single gene

where X_i is the expression level of the *ith* gene of the network, N is the number of genes in the network, $\alpha_i, \beta_i \in \mathbb{R}^N_+$ are *rate constants* and $g_{ij}, h_{ij} \in \mathbb{R}^N$ are *kinetic orders*. It is worth mentioning that the *kinetic orders* g_{ij} and h_{ij} affect the synthesis and degradation of X_i due to X_j.

2.2 RNN as a Model for GRN

Since the inputs for a classic Neural Network are taken iid from the training set, NNs are not suited to model temporal data. The RNN model, however, is a closed loop NN with a delay variable suitable to model dynamic systems (Fig. 1).

The model connects the output of each neuron in the output layer of the RNN to each of the neurons in the input layer via a delay parameter in Fig 1. Vohradský [9] modeled the gene's regulations with this architecture, assuming that each of the neurons in the output unit $(e_i(t+\Delta t))$ is a gene, and the neurons in the input units$(e_{i,\ldots,N}(t))$ are the same genes, thus every gene interacts with one another.

The mathematical model of an RNN resembles a standard NN with additional variables for the feedback loop. The discretized version has the following form:

$$e_i(t+\Delta t) = \frac{\Delta t}{\tau_i} \times f\left(\sum_{j=1}^{N} w_{ij} e_j(t) + \beta_i\right) + \left(1 - \frac{\Delta t}{\tau_i}\right) e_i(t) \tag{2}$$

where, $f()$ is a nonlinear function that acts as a classification function, we use the sigmoid function $f(z) = 1/(1+e^{-z})$. The values w_{ij} are the connecting weights of the network, which represent the connections between gene i and j. The variable e_j represents the expression level for the gene, which is the data we receive from the microarray experiments. And finally, β is the bias parameter of the network.

Forward evaluation evaluates an initial input using the set of weights $W_l = \{w_{ij}\}$ in Eq. 2 and obtains a new time series $e_i(t)$.

Xu et al [3] used PSO to find the weights of an RNN focused on the problem of GRN inference. Here, we used an improved version of PBIL to find the weights of the RNN.

2.3 Estimation Criteria and Regularization

To find the best parameters for the network Tominaga [13] standardized the use of the Mean Squared Error (MSE) evaluation to measure a candidate's fitness in the S-System. They defined the fitness function as:

$$f = \sum_{t=1}^{T} \sum_{i=1}^{N} \left(\frac{X_{i,cal}(t) - X_{i,exp}(t)}{X_{i,exp}(t)} \right)^2 \tag{3}$$

where T represents the number of time samples in the experimental data, N is the number of genes, and *cal*, *exp* refer to the calculated and experimental values of the gene expression's data, respectively. In the case of the S-System, X_{cal} represents the solution of Eq. 1 and for the RNN, it is the output of Eq. 2.

3 Optimization Methods

To find the values that best fit equation and minimize equation 3 using the RNN and S-System models, we use 2 different optimization algorithms from the area of evolutionary computation, which are Dissipative PSO and Population Based Incremental Learning (PBIL). These two methods are described in the following sections and it has to be noted that the described equations are for the general case of its implementation.

3.1 Dissipative PSO

Particle Swarm Optimization (PSO) is an algorithm widely used in different applications. It has been used to analyze human activities [14], optimize power networks [15] and for scheduling problems [16].

In PSO a swarm is composed of particles, each of which can record its best position, and the swarm's best position. Each particle also has a velocity, which updates to grow closer to the best positioned particle in the swarm at each iteration. We define each particle p_i at time $t \in (0, IT-1)$ as $p_i(t) \in \mathbb{R}^N$, where IT is the maximum number of iterations and N is the feature space's dimension. The variables that control each particle at time t are its position $p_i(t)$, and its velocity $v_i(t)$. Each particle's velocity and position will change according to:

$$\begin{aligned} v_i(t+1) = w \cdot v_i(t) + C_1 \cdot \varphi_1 \cdot (P_{li} - p_i(t)) \\ + C_2 \cdot \varphi_2 \cdot (P_G - p_i(t)) \end{aligned} \tag{4}$$

$$p_i(t+1) = p_i(t) + v_i(t+1) \tag{5}$$

where P_{li} is the best position for particle i, P_G is the best position the swarm has obtained and w is the inertia factor, which controls the speed at which the particles adapt. C_1 and C_2 are random variables that control the dependence on the global closeness and the φ_1, φ_2 factors are manually set variables to control the swarm's convergence.

Classic PSO, however, often reaches a local minimum as its final solution. Xiao et al. [4] proposed a variation to the classic PSO, adding a dissipative property to the particles. They defined dissipation parameters that restart the system at random iterations. The dissipative equations, evaluated at each iteration, are:

$$IF(rand() < c_v) => v_i = rand() * v_{max} \tag{6}$$

$$IF(rand() < c_l) => x_{id} = Random(l_d, u_d) \tag{7}$$

where c_v and c_l are numbers between 0 and 1. Setting small numbers to these variables result in few restarts, allowing each new iteration to reach a new optimum. Variables v_{max}, l_d, u_d are the particle's velocity limit, and the lower and upper bound for the search space respectively. The random variables $rand()$ and $Random(I_d, v_d)$ correspond to an uniform random sample and to a sample from the interval given by the lower and the upper bound.

For the S-System, we will define each particle of the swarm with a variable vector p_i, composed of the parameters $\theta_i \in \{g_{ij}, h_{ij}, \alpha_i, \beta_i | i, j \in 1 \dots N\}$.

3.2 PBIL and Clustering

Population Based Incremental Learning (PBIL)[17] is an optimization technique that finds the best candidates of a function by inferring a probability distribution from each of the dimensions in the feature set. This creates N probability distributions, where N is the problem's dimension.

The algorithm chooses the best candidates using the fitness function 3, and then sets a threshold that selects only the best candidates to infer a new probability distribution. New samples will be in turn be taken from this distribution, and a new fitness process will be done. This sampling-fitting process is done recursively until the variations are so low that we will have reached a minimum for the fitness function.

Since the search space is non-convex, if the problem is modeled using standard Gaussians distributions, we are modeling a multimodal problem with an unimodal distribution. Thus conveying local minimum instead of global minimum. In his work, Emmendorfer[18] showed that a mixture of Gaussians, as a multimodal distribution, is a good alternative to model non-convex search spaces.

There are different ways to create a mixture of Gaussians, like doing expectation maximization (EM), or a more naive clustering technique like K-means. K-means are a relaxation of a mixture of Gaussian distributions with symmetric variances, and is a faster procedure than EM for estimation of a mixture of Gaussians.

Using K-means we model the candidates' search space as a mixture of K Gaussian distributions, to have at a set of $N \times K$ clusters modeling the best candidates of the problem for each of the N dimensions.

4 Algorithm to Infer the GRN

The inference algorithm does the following steps:

1. Generate N candidate solutions $X_i \in R^N$ from an uniform distribution $X_i \sim U(X_{min}, X_{max})$.
2. Using the candidates, do the forward evaluation of the RNN or solve the S-System given in equation 2 and 1, respectively, and obtain a set of time series TS per each candidate.
3. Using the cost function (Eq. 3), rank the candidates and choose the best M ones.
4. Update the candidate solutions:
   - (PSO) Using all the candidates, update their values using Eq. 4
   - (PBIL)Using the M candidates, generate $N \times K$ Gaussian distributions using K-means and variance equal to 1.
   - (PBIL)Generate N new candidates from the inferred Gaussian distributions.
5. Go to step 2.

This recursive process is repeated until convergence conditions, which are set by us, are met.

5 Experiments

We tested both models, the S-System with PSO and the RNN with KPBIL, with the popular SOS network for *E.coli* [19]. We ran both of them, and counted the total number of true positives and true negatives to calculate measurement variables like Recall, Precision and F-Score.

To test the effect of the variables in the PBIL, we changed the population size, from 100 to 500 candidates, as well as the cluster number in the K-means, from 1 to 5, to test whether multiple clusters do better than a single probability density. For the PBIL implementation, we did 2000 iterations per run, with each iteration lasting at most 5 minutes for the architectures with 500 candidates.

In PSO, we evaluated the effect of the population as well, and variated it from 20 to 320. We did 5000 iterations per run, with each of them evolving 20 particles. Each of the iterations took 3.2 hours to reach a steady state.

To evaluate the variables, we used the Recall, Precision, True Negative Rate (1-Specificity(S_p)) and F-Score using the following equations:

$$Recall = \frac{TP}{TP+FN} \qquad Precision = \frac{TP}{TP+FP}$$

$$S_p = \frac{TN}{TN+FP} \qquad F-Score = 2\frac{Precision * Recall}{Precision + Recall}$$

where TP, FP, TN and FN stand for True and False positive and negative, respectively. In our approach, a true positive is assigned when a connection between 2 genes is truth and a true negative, when the absence of the connection is true.

5.1 Real SOS Network

The SOS network[19] for *E.coli*, published by the Uri Alon group (Fig. 2), is a benchmark for testing genetic regulation inference.

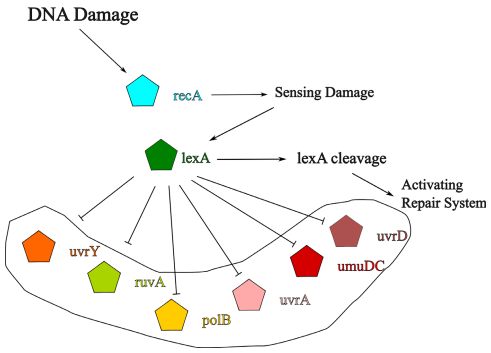


Fig. 2. Graphical Representation of the SOS Net, where the lexA represses every gene

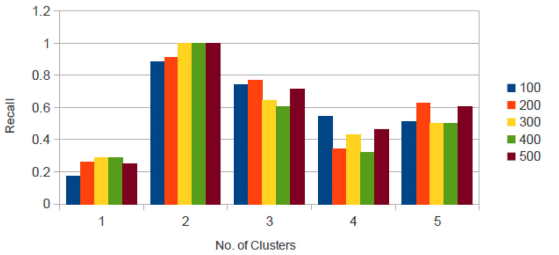
In their experiments, 8 genes are expressed (uvrD, lexA, umuD, recA, uvrA, uvrY, ruvA and polB). They irradiate the DNA with UV light, which affects some genes, after that, the network will repair itself. They did four experiments for different light intensities. Each experiment had 50 time steps spaced by 6 minutes. Many researchers, however, usually choose to infer only 6 of the 8 genes, since two of them have marginal activity in comparison with the rest of the genes in the network. For the sake of comparison, we have worked also exclusively with 6 genes.

5.2 Comparative Analysis on the Population Size and Number of Clusters

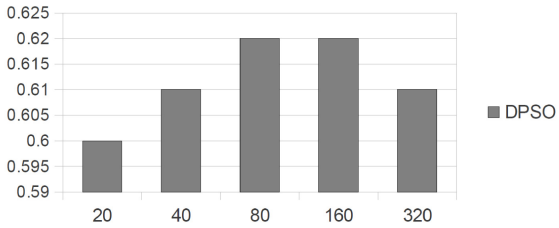
We plotted the Recall for different population sizes and number of clusters in Figs. 3a and 3b.

Fig. 3a shows that for few clusters, the recall is good, and generally, large populations present marginally better results than small populations. This is because the K-means method does not require many particles to create reliable clusters, furthermore, clustered approaches have better results than non-clustered approaches. For this problem, then, using more than 3 clusters seems to dampen the search, and a few clusters —2 or 3— is the best alternative. We can see as well that the unclustered option —1 cluster— has worst results than its alternatives,

Fig. 3b shows that for large populations the recall of the DPSO is reduced by a small factor, this is because we let them run for the same time. Intuitively, large populations will spend more resources than small populations, so it takes longer to converge to a good solution. The results, however, show that the DSPO algorithm is capable of obtaining good results for a small number of particles.



(a) Recall for Different Clusters and Population for the KPBIL. Evaluation for 100 to 500 possible candidates.



(b) Recall values for different population sizes using DPSO

Fig. 3. Recall mean and standard deviation for the SOS Network

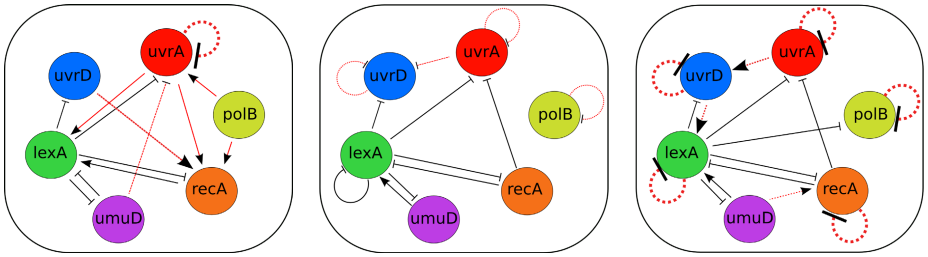
In Fig. 4 we show a qualitative solution to the problem of finding the right connections in the SOS Network. Fig. 3b shows the result of using DPSO, which has many false positives, related with the low recall that we presented before.

In Fig. 4b we see the inferred network using a traditional PBIL, without any clustering to infer multiple probabilities distributions. While the performance was better than the DPSO, it still finds many false positives, specially on self regulations.

Finally, Fig. 4c presents the results using the KPBIL approach, which has the most promising results of them three. The network however becomes sparse, losing some important connections.

To compare our results, we compiled results from other papers working with the SOS Net. Table 1 shows this compilation, here as well we have a quantitative comparison of our approach.

Table 1 shows that KPBIL does inference better than other approaches that use both evolutionary and non-evolutionary techniques. KPBIL is much faster than similar approaches, specially comparing with the state of the art, which is the S-Tree, which takes 35 hours to do the whole inference. DPSO presents one of the best sensitivities of the group, this is because the S-System iw one of the most precise models for these kind of problems, while the RNN model is just an approximation that is used for the sake of speed but at the expense of precision.



(a) DPSO & S-System (b) PBIL & RNN with one cluster (c) KPBIL & RNN with 3 clusters

Fig. 4. Solutions for different combinations of PBIL and DPSO, the straight lines are True connections while the red dotted lines are spurious connections

Table 1. Specificity, Sensitivity and F-Score for different models of the *E.coli* SOS Network

	#Regs	#TP	#FP	#TN	#FN	Sensitivity	Specificity	F-Score	Time[h]
Bayesian Network[20]	6	4	2	18	3	0.571	0.900	0.615	0.01
S-Tree[21]	7	6	1	19	1	0.857	0.950	0.857	35
LTV[22]	13	7	6	14	0	1.000	0.700	0.7	0.1
DE[1]	8	5	3	17	2	0.714	0.850	0.667	0.3
KPBIL	11	7	4	13	3	0.7	0.765	0.67	0.05
DPSO	10	7	3	17	0	1	0.68	0.60	3.5

6 Conclusions

We have presented a comparison between two methods to do inference in Gene Regulatory Networks, a Recursive Neural Network with PBIL and the S-System using DPSO. We have done comparison and analysis of both methods using the SOS network for *E.coli*, which is a benchmark for small network inference. We presented in the results that both RNN and PBIL present promising results for the inference problems, it had both better results and faster inference rate, which are two of the most important desiderata in molecular biology.

For future work, we will attempt the use of our model for larger networks, modeling larger networks remains a challenge for this kind of approaches, since the complexity increases vastly with each new gene we add to the system.

References

1. Noman, N., Iba, H.: Inferring gene regulatory networks using differential evolution with local search heuristics. IEEE/ACM Transactions on Computational Biology and Bioinformatics 4(4), 634–647 (2007)
2. Kikuchi, S., Tominaga, D., Arita, M., Takahashi, K., Tomita, M.: Dynamic modeling of genetic networks using genetic algorithm and S-system. Bioinformatics 19(5), 643 (2003)
3. Xu, R., Donald Wunsch, I.I., Frank, R.: Inference of genetic regulatory networks with recurrent neural network models using particle swarm optimization. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 681–692 (2007)
4. Xie, X.F., Zhang, W.J., Yang, Z.L.: Dissipative particle swarm optimization. In: Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002, vol. 2, pp. 1456–1461. IEEE (2002)
5. Akutsu, T., Miyano, S., Kuhara, S.: Identification of genetic networks from a small number of gene expression patterns under the Boolean network model. In: Pacific Symposium on Biocomputing, vol. 4, pp. 17–28. Citeseer (1999)
6. Murphy, K., Mian, S.: Modelling gene expression data using dynamic Bayesian networks. Graphical Models, 12 (1999)
7. Noman, N., Iba, H.: Inference of gene regulatory networks using s-system and differential evolution. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, Washington, DC, p. 439. Citeseer (2005)
8. Kimura, S., Ide, K., Kashihara, A., Kano, M., Hatakeyama, M., Masui, R., Nakagawa, N., Yokoyama, S., Kuramitsu, S., Konagaya, A.: Inference of S-system models of genetic networks using a cooperative coevolutionary algorithm. Bioinformatics 21(7), 1154 (2005)
9. Vohradský, J.: Neural network model of gene expression. The FASEB Journal: official publication of the Federation of American Societies for Experimental Biology 15(3), 846–854 (2001)
10. Pettersson, F., Biswas, A., Sen, P.K., Saxén, H., Chakraborti, N.: Analyzing Leaching Data for Low-Grade Manganese Ore Using Neural Nets and Multiobjective Genetic Algorithms. Materials and Manufacturing Processes 24(3), 320–330 (2009)
11. Zamparelli, M.: Genetically Trained Cellular Neural Networks. Neural Networks: The Official Journal of the International Neural Network Society 10(6), 1143–1151 (1997)
12. Savageau, M.A.: Biochemical systems analysis+*:: I. Some mathematical properties of the rate law for the component enzymatic reactions. Journal of Theoretical Biology 25(3), 365–369 (1969)
13. Tominaga, D., Koga, N., Okamoto, M.: Efficient numerical optimization algorithm based on genetic algorithm for inverse problem. In: Proceedings of the Genetic and Evolutionary Computation Conference, vol. 251, p. 258 (2000)
14. Palafox, L., Hashimoto, H.: 4W1H and Particle Swarm Optimization for Human Activity Recognition. Journal of Advanced Computational Intelligence and Intelligent Informatics 15(7), 793–799 (2011)
15. AlRashidi, M., El-Hawary, M.: A survey of particle swarm optimization applications in electric power systems. IEEE Transactions on Evolutionary Computation 13(4), 913–918 (2009)
16. Liao, C., Luarn, P.: A discrete version of particle swarm optimization for flowshop scheduling problems. Computers & Operations Research 34(10), 3099–3111 (2007)

17. Sebag, M., Ducoulombier, A.: Extending population-based incremental learning to continuous search spaces. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 418–427. Springer, Heidelberg (1998)
18. Emmendorfer, L., Pozo, A.: Effective Linkage Learning Using Low-Order Statistics and Clustering. IEEE Transactions on Evolutionary Computation 13(6), 1233–1246 (2009)
19. Ronen, M., Rosenberg, R., Shraiman, B.I., Alon, U.: Assigning numbers to the arrows: parameterizing a gene regulation network by using accurate expression kinetics. Proceedings of the National Academy of Sciences 99, 10555 (2002)
20. Perrin, B.E., Ralaivola, L., Mazurie, A., Bottani, S., Mallet, J., D'Alche-Buc, F.: Gene networks inference using dynamic Bayesian networks. Bioinformatics 19(suppl. 2), ii138–ii148 (2003)
21. Cho, D.Y., Cho, K.H., Zhang, B.T.: Identification of biochemical networks by S-tree based genetic programming. Bioinformatics 22, 1631–1640 (2006)
22. Kabir, M., Noman, N., Iba, H.: Reverse engineering gene regulatory network from microarray data using linear time-variant model. BMC Bioinformatics 11(suppl. 1), S56 (2010)