Gene networks reconstruction and time-series prediction from microarray data using recurrent neural fuzzy networks

I.A. Maraziotis, A. Dragomir and A. Bezerianos

Abstract: Reverse engineering problems concerning the reconstruction and identification of gene regulatory networks through gene expression data are central issues in computational molecular biology and have become the focus of much research in the last few years. An approach has been proposed for inferring the complex causal relationships among genes from microarray experimental data, which is based on a novel neural fuzzy recurrent network. The method derives information on the gene interactions in a highly interpretable form (fuzzy rules) and takes into account the dynamical aspects of gene regulation through its recurrent structure. To determine the efficiency of the proposed approach, microarray data from two experiments relating to *Saccharomyces cerevisiae* and *Escherichia coli* have been used and experiments concerning gene expression time course prediction have been conducted. The interactions that have been retrieved among a set of genes known to be highly regulated during the yeast cell-cycle are validated by previous biological studies. The method surpasses other computational techniques, which have attempted genetic network reconstruction, by being able to recover significantly more biologically valid relationships among genes.

1 Introduction

The investigation of complex biological processes at the molecular level is now possible by monitoring the gene expression activity at the whole genome level [1]. The amount and the complexity of information that gene expression data contains, together with the inherent large dimensionality of available data sets, poses new challenges to the data analysis research community and thus requires novel data analysis and modelling techniques. The availability of expression data initially made possible the inference of functional information for genes of unknown functionality in several partially mapped genomes by means of co-expression clustering [2, 3]. Other approaches based on supervised learning techniques have resulted in the development of novel diagnosis tools that discriminate between different sample classes (e.g. healthy against diseased tissue diagnosis) [4]. However, as the ultimate goal of molecular biologists is to uncover the complex regulatory mechanisms controlling cells and organisms, the reconstruction and modelling of gene networks remains one of the central problems in functional genomics.

Proteins and metabolites, which are produced by proteins, regulate the activity of genes. Proteins, however, are also gene products and thus genes can influence each other (induce or repress) through a chain of proteins and metabolites. At the genetic level, it is thus legitimate, and indeed common, to consider gene–gene interactions, and these lead to the concept of gene networks. Gene networks ultimately attempt to describe how genes or groups of genes interact with each other and identify the complex regulatory mechanisms that control the activity of genes in living cells. The reconstructed gene interaction models should be able to provide biologists with a range of hypotheses explaining the results of experiments and suggesting optimal designs for further experiments.

The reconstruction of gene networks based on expression data is hampered by peculiarities specific to this kind of data; therefore the methods employed should be able to handle underconstrained data, should be robust to noise (as experimental data obtained from microarrays are measurement noise-prone), and should be able to provide interpretable results. Recently, there have been several attempts to describe models for gene networks. Boolean networks have been used due to their computational simplicity and their ability to deal with noisy experimental data [5]. However, the Boolean network's formalism assumes that a gene is either on or off (no intermediate expression levels allowed) and the models derived have inadequate dynamic resolution. Other approaches, reconstructing models using differential equations, have proved to be computationally expensive and very sensitive to imprecise data [6]. Models based on Bayesian networks, although attractive due to their ability to deal with stochastic aspects of gene expression and noisy measurements, have the disadvantage of minimising the dynamical aspects of gene regulation [7].

Our approach uses a novel recurrent neural fuzzy network to extract information from time-series gene expression data from microarray experiments. It is known that both fuzzy logic systems and neural networks aim at exploiting a human-like knowledge processing capability. Artificial neural networks are well-known universal function approximators, in terms that they can approximate any continuous function represented by data subject to having an appropriate neural

doi:10.1049/iet-syb:20050107

Paper first received 21st December 2005 and in revised form 9th April 2006

The authors are with the Department of Medical Physics, Medical School, University of Patras, Rio 26500, Greece

E-mail: imarazi@heart.med.upatras.gr

network model. A recurrent neural network naturally involves dynamic elements in the form of feedback connections used as internal memories. Unlike the feedforward neural network whose output is a function of its current input only and is limited to static mapping, recurrent neural networks perform dynamic mapping. In contrast, fuzzy logic is a natural language for linguistic modelling; as a consequence, it is consistent with the qualitative linguistic–graphical methods used to describe biological systems. Neuro-fuzzy systems combine the advantages of computational power and low-level learning common to neural networks, and the high-level human-like reasoning of fuzzy systems. The dynamic aspects of gene regulatory interactions are considered by the recurrent structure of the neuro-fuzzy architecture we propose, while the online learning algorithm drastically reduces the computational time.

The method was experimentally validated by applying it to two real biological data sets from *Saccharomyces cerevisiae* (yeast) and *Escherichia coli*, from where we prove experimentally the neural fuzzy recurrent network (NFRN) ability for gene time-series prediction. Interactions among a set of target genes found to be highly involved in the yeast cell-cycle regulation from previous biological studies [8] are studied and knowledge in the form of IF–THEN rules is inferred. The algorithm is trained on a subset of experimental samples and the inferred relations are tested for consistency on the remaining samples. We demonstrate that the proposed approach manages to single out the presence of gene regulatory relations, which is not apparent when other methods are applied.

2 Materials and methods

The computational approach described in this paper is that of a multilayer NFRN. In the literature, there exist several recurrent neural fuzzy network models such as D-FUNCOM [9], TRFN [10] and RSONFIN [11]. Being a hybrid neuro-fuzzy architecture, it is able to overcome the drawbacks specific to pure neural networks, which function as black boxes. By incorporating elements specific to fuzzy reasoning processes, we are able to give to each node and weight their meaning and function as part of a fuzzy rule, instead of just being abstract numbers.

In addition, its recurrent structure manages to possess the same advantages of a pure recurrent neural network (in terms of computational power and time prediction), while succeeding in extending the application of the classic neuro-fuzzy networks to temporal problems. In the following, we illustrate our technique for using a number of NFRN models to extract relations in the form of fuzzy IF–THEN rules that describe a gene network.

2.1 Neural fuzzy recurrent network

NFRN adopts the Zadeh–Mamdani's fuzzy model [12] to realise fuzzy inference and each one of the rules for the case of multi-input multi-output has the following form

$$R^i\text{: If } x_1 \text{ is } A_{1i} \text{ and } x_2 \text{ is } A_{2i} \text{ and } \cdots \text{ and } x_n \text{ is } A_{ni}$$
$$\text{Then } y_1 \text{ is } B_{1i} \text{ and } \cdots \text{ and } y_m \text{ is } B_{mi} \qquad (1)$$

where $A_{1i}, A_{2i}, \ldots, A_{ni}$ and $B_{1i}, B_{2i}, \ldots, B_{mi}$ are fuzzy sets that are used from the N_R ($i = 1, 2, \ldots, N_R$) fuzzy IF–THEN rules to perform a map from the input space, which has the form of a vector $\boldsymbol{X} = [x_1, x_2, \ldots, x_n]^{\mathrm{TM}} P^n$ to the output space vector $\boldsymbol{Y} = [y_1, y_2, \ldots, y_m]^{\mathrm{TM}} P^m$.

In contrast to other neuro-fuzzy network architectures, where the network structure is fixed and the rules should be assigned in advance, there are no rules initially in the architecture we are presenting; all of them are constructed during online learning. Two learning phases, the structure as well as parameter learning phases, are used to accomplish this task. The structure learning phase is responsible for the generation of fuzzy IF–THEN rules as well as the judgement of the feedback configuration, and the parameter learning phase for tuning the free parameters of each dynamic rule (such as the shapes and positions of membership functions), which is accomplished through repeated training on the input–output patterns.

The way the input space is partitioned determines the number of rules. Given the scale and complexity of the data, the number of possible rules describing the causal relationships is kept under constraint by employing an aligned clustering-based partition method for the input space, meaning that both input and output variables may have a different number of fuzzy sets describing them, and by allowing a scene in which rules with different preconditions may have the same consequent part [11]. By incorporating the above clustering scheme, we are able to tackle the problem of rule set combinatorial explosion that appears when using other neuro-fuzzy approaches like ANFIS. One of the main issues this paper attempts to address is the extraction of gene networks in the explanatory form of IF–THEN rules. In order for this to be feasible, the rules must be both meaningful and easily interpretable; thus we have to restrict the number of fuzzy sets on every variable to a maximum of seven, even though this might lead to slightly less accurate results. From our empirical observations, further increasing the number of fuzzy sets does not provide additional insightful information but unnecessarily increases the complexity. In contrast, experimentation has proved that an attempt to keep the number of fuzzy sets less than three fails to give adequate resolution.

2.1.1 NFRN architecture: The architectural structure of NFRN is described in Fig. 1. The network consists of six layers. The dimension of the first layer matches the number of the input variables, while the dimension of the sixth layer equals the number of output variables. The number of nodes in the fourth layer equals the number of rule nodes, while the dimensions of the remaining layers are not known in advance, as they are created automatically by the structure learning algorithm of the NFRN. In the following, we use the symbol $\varphi_i(k)$ to denote the input of node i in the kth layer while the symbol $\psi_i(k)$ will denote the output of node i in the kth layer.

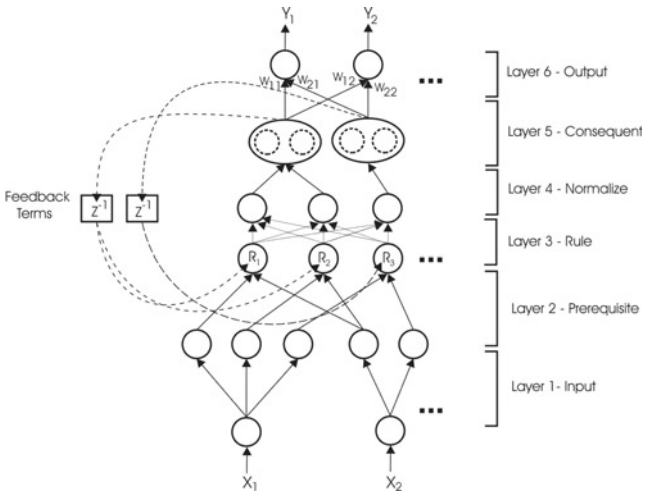

Fig. 1 *Architectural structure of the proposed NFRN*

The nodes in the first layer represent an input variable. There is no computation in the first layer and the values are transmitted directly to the next layer

$$\psi_i^{(1)} = \varphi_i^{(1)} = x_i \tag{2}$$

Each node in the second layer corresponds to a linguistic label (e.g. low-expressed, average-expressed highly-expressed etc.) and is represented by a membership function of the Gaussian form

$$\psi_{ij}^{(2)} = \exp\left\{-\frac{(x_i - c_{ij})^2}{\sigma_{ij}^2}\right\} \tag{3}$$

where i runs through all the input variables and j runs through the number of fuzzy sets of each one of the i input variables, ψ_{ij} is the jth membership function of the ith input variable, c_{ij} and σ_{ij} are the centre and width of the ψ_{ij} membership function, while x_i is the value of the ith input variable. The linguistic labels result from the fuzzification of the expression data, which translates continuous attributes into fuzzy ones, efficiently managing the uncertainty and the vagueness of the expression levels. Qualitative descriptors such as 'highly-expressed', 'average-expressed', 'low-expressed' and so on are employed to denote values of expression data. Each node in this layer calculates the membership value specifying the degree to which an input value belongs to a linguistic label. This is accomplished by comparing the input value with the centre c and width σ of the corresponding fuzzy sets for every linguistic label. Other types of membership functions like trapezoidal or triangular could be used.

The nodes of the third layer are called rule nodes. Every rule node represents a fuzzy logic rule and performs pre-condition matching for the rule. The following fuzzy AND operation is performed by all nodes in this layer

$$\psi_i^{(3)}(t) = \beta_i \cdot \psi_j^{(5)}(t-1) \cdot \prod_k \phi_i^{(3)}(t) \tag{4}$$

while we have that

$$\prod_k \phi_i^{(3)} = \exp\left\{-[D_k(x_k - c_k)]^{\mathrm{T}}[D_k(x_k - c_k)]\right\} \tag{5}$$

where the value of i is between 1 and the number of rule nodes, k runs through all the nodes of the second layer that are inputs to the ith rule node, and j runs through all the nodes of the fifth layer that serve as consequents of the ith rule. Also $D_i = \mathrm{diag}(1/\sigma_{i1},\ 1/\sigma_{i2}, \ldots,\ 1/\sigma_{in})$, $c_i = (c_{i1},\ c_{i2}, \ldots,\ c_{in})^{\mathrm{T}}$ and β_i denotes the link weight from a feedback unit to a rule node. As we can see from (4), the input for each node of the layer comes from two sources. The first is from layer 2 and the second from layer 5 where the former represents the rule's spatial firing degree and the later the rule's temporal firing degree by means of the memory terms $\psi_j^{(5)}(t-1)$.

Layer 4 is the normalisation layer. The number of nodes in this layer is equal to that of layer 3. The outputs of the nodes of the previous layer are normalised in this layer by the following operation

$$\psi_i^{(4)} = \frac{\phi_i^{(4)}}{\sum_j \phi_j^{(4)}} \tag{6}$$

where j runs through all the rule nodes.

The fifth layer is responsible for the segmentation of the output space

$$\psi_i^{(5)} = \sum_i \varphi_i^{(5)} \tag{7}$$

The equation states that the output of each of the nodes of the current layer is the sum of the output of the rule nodes that have as consequent part the current node.

Nodes in the last layer correspond to output linguistic variables. In this layer, we have the realisation of the defuzzification operation which is performed by

$$y_j = \psi_i^{(6)} = \sum_i w_{ij}\varphi_i^{(6)} \tag{8}$$

where w_{ij} is the link weight between the nodes of layer 5 and layer 6.

2.1.2 Partitioning of input–output space: Following, we provide a detailed description of the learning algorithm of NFRN, presented in a flow-chart form in Fig. 2. The algorithm's first phase is concerned with the structure

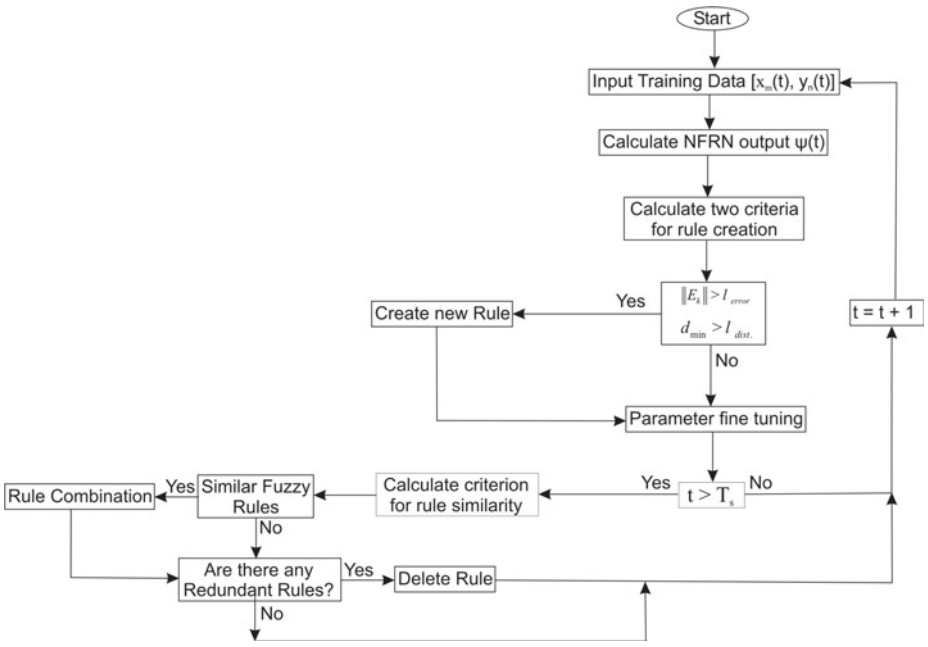

Fig. 2 *Structure of the NFRN learning algorithm*

learning of the network, which is directly connected with the partitioning of the input and output space. The creation of a new rule corresponds to the creation of a new cluster in the input space. Therefore the way the input space is partitioned–clustered determines the number of fuzzy rules created. This fact lead us to the conclusion that the number of rules created by NFRN is problem dependent, that is the more complex a problem is, the greater the number of rules becomes. A similar scheme stands for the output space, as the number of generated clusters in the consequent space is also problem dependent. The number of output clusters is large for complex problems and small for simple ones. In order for the NFRN to decide whether a new rule must be generated for the description of an incoming pattern $(\boldsymbol{x}, \boldsymbol{y})$, a two criteria scheme is adopted. The first one computes the overall error E_k, which is defined as the difference between the actual and the desired output and is given by

$$\|E_k\| = \|y_k^d - y_k\| \tag{9}$$

where y_k is the value computed by the model based on the current input pattern $\boldsymbol{X}_k$ while y_k^d is the desired output for the same pattern.

The second criterion is based on the calculation of the distance d_i between the newly arrived observation $\boldsymbol{X}_k$ and all of the rules that have been created up to time step k

$$d_k(j) = \|\boldsymbol{X}_k - R_j\|, \quad j = 1, 2, \ldots, N_r \tag{10}$$

where N_r is the number of the rules, each one of the d_k is computed by using equation (5). Then we find

$$d_{\min} = \min_j(d_k(j)) \tag{11}$$

Now, if both the criteria below are true, we create a new rule

$$\begin{aligned} \|E_k\| &> \lambda_{\text{error}} \\ d_{\min} &> \lambda_{\text{dist}} \end{aligned} \tag{12}$$

The first criterion checks whether the error of the model is greater than a specific value, which shows that in the current state of the network we cannot deal with the new pattern without a large value of the error. The second checks to see whether the pattern is 'close' enough to an existing cluster, so that it can become a member of it, or if a new cluster has to be created for it.

If the procedure described leads to the creation of a rule, the next step is the assignment of initial centres and widths of the corresponding membership functions. Given the fact that the centres and widths will be refined later in the parameter learning scheme, we simply set

$$c_i(t+1) = x_i \tag{13}$$

$$\sigma_i(t+1) = -\frac{1}{\delta}\frac{1}{\ln(d_{\min})} \tag{14}$$

where i runs through all the input variables, δ is a constant deciding the overlap of the clusters, $d_{\min}$ is the distance to the closest cluster (from the current pattern) and c_i, σ_i are the centre and width for the membership function of each input variable, respectively.

A similar method, where width σ is accounted for in the degree measure for the creation of a new radial basis unit, has been used before [9, 11]. This method ensures that for the description of a cluster with larger width (which means a larger covering region), fewer rules will be generated. To reduce the number of fuzzy sets of each input variable and to avoid the existence of redundant fuzzy sets, we check the similarities between them in each input dimension. For the similarity measure of two fuzzy sets, we use the formula previously derived in the work of Lin and Lee [13], which concerns bell-shaped membership functions.

Finally, NFRN has to determine if a new output cluster must be created. A cluster in the output space is the consequent of a rule. We have already seen that one of the characteristics of NFRN is that more than one rule can be connected to the same consequent. As a result, the creation of a cluster in the input space does not necessarily mean a subsequent creation of a cluster in the output space; it depends on the incoming pattern because the newly created cluster in the input space could be connected to an already existing cluster of the output space.

The model decides whether or not to create a new output cluster based on the two criteria described earlier (12).

2.1.3 Merging and deletion of rules: As a concluding step in the structure learning process of NFRN, we have developed a scheme for deleting redundant rules and combining similar fuzzy rules into an equivalent new one. This process has the dual goal of both decreasing the redundancy of the model as well as increasing the model's simplicity.

If a fuzzy set is near zero over its own universe of discourse for a certain number of time steps, then the rule with the specific fuzzy set as a precondition should be removed, as this fact indicates that the output of the rule is also near zero.

When two fuzzy rules have different consequents but very similar antecedents, we have a case of conflict among those rules. The solution to this conflict problem is either to delete one of the rules, or combine them into a new one. Therefore we decide whether we can combine two rules by evaluating the similarity of their antecedent part. Given, for example, two rules R_i and R_j, their antecedent parts are $A_{i1}, A_{i2}, \ldots, A_{in1}$ and $A_{j1}, A_{j2}, \ldots, A_{jn1}$, respectively, where A_{kl} is the kth fuzzy set of the lth fuzzy rule and the similarity for the antecedents can be computed as

$$S(A_i, A_j) = \min_l\{S(A_{il}, A_{jl})\}, \quad l = 1, 2, \ldots, n1 \tag{15}$$

if $S(A_i, A_j)$ exceed a value l_s then those fuzzy sets are considered to be similar and thus the rules that they constitute can be combined towards the creation of a new rule R_c. For the evaluation of the centre and width of the joined fuzzy sets A_{cl}, we use the average of the fuzzy sets A_{il}, A_{jl}, $c_{cl} = (c_{il} + c_{jl})/2$, $w_{cl} = (w_{il} + w_{jl})/2$. The same technique is used for the two consequents, meaning $z_c = (z_i + z_j)/2$. Here again, we follow the same methodology used by Lin and Lee [13] for checking the similarity among two fuzzy sets.

Usually the main reason for a case where we have to combine rules is the inherent noise in the microarray input data. Another reason though for such a scenario is that there might be a gap in the experimental knowledge and that triggers us to mark those genes in order for future experiments to specify this behaviour. A measure for distinguishing among those two grounds is the value of l_s.

It should be pointed out that the value of l_s has an initial value that decays through time so that higher similarity between two fuzzy sets is allowed in the initial stage of learning.

2.1.4 Parameter learning scheme: After the structure of the network is completed according to the current training pattern, the NFRN enters the parameter learning scheme to adjust the parameters of the network (e.g. widths and centres of the fuzzy sets) based on the same

training pattern. At this point, we should state that, as can be noted, the parameter learning process is performed concurrently with the structure learning. For the parameter identification of the NFRN, we have used an algorithm based on the method of gradient descent. The backpropagation learning algorithm is a widely used algorithm for training neural networks or fuzzy networks by means of error propagation via variation calculus. Owing to the simplicity of the backpropagation through time (BPTT) learning algorithm, we adopt this method to tune free parameters of the NFRN model. This study attempts to emphasise the methodology and dynamic mapping abilities of the NFRN model in the subject of gene network reconstruction based on microarrays time series expression data. Lee and Teng [14] also adapted BPTT to successfully train the recurrent fuzzy structure they proposed. Of course, there are many existing learning algorithms in the literature like the real-time recurrent learning algorithm [15] or the order-derivative algorithm [16] for training recurrent neural networks that can also be adapted for the fine tuning of the NFRN model.

2.2 NFRN models for the description of gene networks

A method for depicting gene regulatory networks is now described, through fuzzy IF–THEN rules, with the help of the NFRN model described in the preceding section.

In order to recover functional relationships among genes and to build models describing their interactions, we employ transcriptional data from microarray experiments. A simple representation of the data would be that of a matrix with rows containing gene expression measurements over several experimental conditions/samples. Gene expression patterns give an indication of the levels of activity of genes in the tissues under study. A number of network models that match the number of genes are constructed, each model having as an output node one selected gene and as input nodes the remaining genes from the data set. Models attempt to describe the behaviour of the gene selected as output based on possible interactions from the input genes and stores the knowledge on the derived gene–gene interactions in a pool of fuzzy rules.

For each model, we use an error criterion, in order to test the accuracy on a given data set, by using an overall error measure based on the difference between the predictions of the model for the output variable (i.e. gene). Several criteria exist to fulfil this purpose; two of the most common are mean square error and mean error, described by the following equations

$$E_i = \frac{1}{N}\sum_{i=1}^{N}\left(y_i^{\text{predicted}} - y_i^{\text{measured}}\right)^2 \tag{16}$$

$$E_i = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|y_i^{\text{measured}} - y_i^{\text{predicted}}\right|}{y_i^{\text{measured}}} \tag{17}$$

where N is the number of samples in the data set.

The number of rules governs the expressive power of the network. Intuitively, an increase in the number of rules describing the gene interactions in a certain model results in a decrease in the error measure of (16) accounted for by the respective model. However, pursuing a low error measure may result in the network becoming tuned to the particular training data set and exhibiting low performance on the test sets (the well-known overfitting problem). Therefore the problem becomes one of combinational optimisation: minimise the number of rules while preserving the error measure at an adequate level by choosing an optimal set of values for the model parameters (overlap degree between partition clusters, learning rates, rule creation criteria).

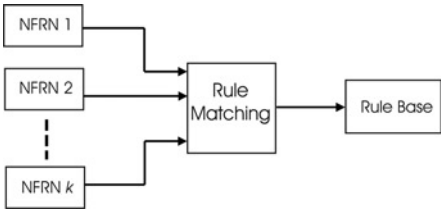

Fig. 3 *Rule base extraction*

Final rule base is formed by a set of rules consistent with all the individual NFRN network models

The final rule base extracted from a certain data set, which provides the plausible hypotheses for gene interactions, is built by selecting the rules that are consistent with all the models (Fig. 3).

3 Results

In the experimental analysis, the methodology we used foresees as a first step the use of NFRN models for time series prediction of gene expression, and as a second step, the rules created by these models employed in order to recover gene regulatory network structures. For this second step, we extracted a part of the yeast cell cycle pathway represented in the KEGG database and used it for the inference experiment. The target network is depicted in Fig. 4. In this section, we also describe the biological data sets we have used for the conducted experiments.

3.1 Data

To test the validity of our approach, we have used two data sets originating from microarray experiments. The first data set contains gene expression measurements during the cell cycle of the budding yeast. The experiments were performed by Spellman *et al.* [17] and consisted of 59 samples collected at different time points of the cell cycle. The experiments were divided into three subsets, which were named according to the synchronisation method used for the yeast cultures: *cdc15 arrest* (24 samples), *cdc28 arrest* (17 samples) and *αlpha-factor* (18 samples). Missing values were filled in using an estimation method based on Bayesian principal component analysis, which estimates simultaneously a probabilistic model and latent variables within the framework of Bayes inference [18].

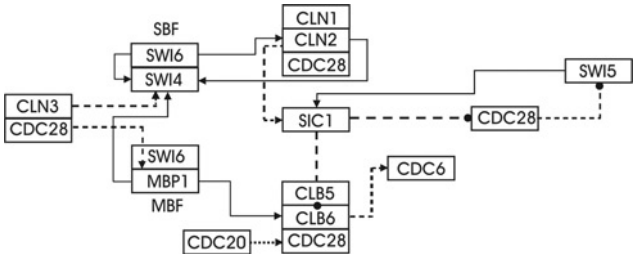

Fig. 4 *Schematic representation of the known yeast cell cycle (based on the KEGG database) interactions among protein products for the genes we study*

Arrows and circles represent positive and negative interaction, respectively

Dashed line indicates a protein–protein interaction, and solid lines indicate direct transcriptional regulation

The second data set we used contained gene expression measurements presented by Ronen *et al.* [19] for the SOS DNA Repair network of *E. coli* bacterium. The specific study consisted of four experiments under various light intensities (experiment 1 and 2: 5 J m^{-2}, experiment 3 and 4: 20 J m^{-2}). Each experiment consisted of 50 time points with a time period of 6 min in between, while eight major genes were monitored: uvrD, lexA, umuD, recA, uvrA, uvrY, ruvA and polB. The specific data set can be downloaded at http://www.weizmann.ac.il/mcb/UriAlon/Papers/SOSData in the form of four different data sets, corresponding to Exp. 1, Exp. 2, Exp. 3 and Exp. 4, respectively.

3.2 Experimental analysis

We chose a subset of 12 genes from yeast (Table 1) on the basis that they were identified in previous biological studies to be highly regulated during the yeast cell cycle, their protein products playing key roles in controlling the cyclic biological processes [8]. The samples of the *cdc15* subset were used as training data for the neuro-fuzzy network and the inferred fuzzy rules were tested for consistency on the other two experimental data subsets (*alpha* and *cdc28*). The choice of the training set was determined by the larger number of samples in the *cdc15* experimental set and therefore a larger number of training instances.

We employed the recurrent neuro-fuzzy approach to create 12 multi-input, single-output models. Each model describes the state of an output gene based on the temporal expression values of the remaining 11 genes. As already described in the methods section, the structure of each model is determined by the online operation of the algorithm using the *cdc15* subset as a training set and repeated training of the algorithm is subsequently used to fine tune the membership functions as well as to determine the precondition and consequent parts of the rules.

Each constructed network describes the time series of the output gene through a number of fuzzy rules. Table 1 presents the set of rules describing gene SWI4. Columns describe rules. The numbers correspond to membership function values for each input variable – gene, expression level, that is 3 is high, 2 is medium and 1 is low (e.g. the second column contains the rule: 'If SIC1 is low AND CLB5 is medium AND ... AND MBP1 is high AND CDC6 is low THEN SWI4 is medium').

We used the network models derived from the training set to predict time series for the gene expression patterns. Table 2 presents the prediction mean square errors on all three data subsets [computed as in (16)], while Fig. 5 shows the predicted time series for the expression ratio of genes SIC1 and CLN3 on both test data sets. As a perfect fit is given by a zero error, most of the models (as the one corresponding to gene CLN3) yield low error rates, managing to consistently and accurately fit the real expression data. Nevertheless, the error values presented in the table are just comparative measures for the performance of the constructed models, an indicative measure being given by the fact that even models yielding high error rates in the table (as is the case of the model corresponding to SIC1) still manage to perform accurate prediction of the expression patterns.

At this point and prior to continuing our analysis, we would like to point out that in neural networks, one of the major pitfalls is overtraining, otherwise known as overfitting. Overtraining occurs when a network has learned not only the basic mapping associated with input and output data, but also the subtle nuances and even the errors specific to the training set. If too much training occurs, the network memorises the training set and loses its ability to generalise new data. The result is a network that performs well on the training set but performs poorly on out-of-sample test data and later during actual trading. There are many methods used to avoid overtraining like stopping training early, regularisation, weight decay, adding noise or jittering and the use of a validation data set. In order to address this issue, we performed a second test in an attempt to further check our method against overfitting. We created an artificial data set based on the same original data (in this way, keeping the characteristics of it), by adding random noises at each measurement of the data set. Normal variations with mean zero and standard deviation 0.5 were used. In this experimental setup, we used a well-known method for checking overfitting called 3-fold cross-validation. In this scheme, we repeatedly used two out of the three data sets (alpha, cdc15, cdc28) as training data set and the remaining set was used as a validation-testing data set. The results of the experiment in terms of the MSE metric, are shown in Table 3. Notice that the models manage to get very close

Table 1: Rules describing the state of gene SWI4 based on the 11 remaining genes of the original data set *cdc15*

Input genes	Rules		
	Rule 1	Rule 2	Rule 3
SIC1	3	1	2
CLB5	3	2	1
CDC20	2	1	3
CLN3	3	1	2
SW16	2	3	1
CLN1	3	2	1
CLN2	3	2	1
CLB6	3	2	1
CDC28	2	1	3
MBP1	2	3	1
CDC6	3	1	2
Prediction of SWI4	3	2	1

Each rule is read per column and the numbers represents an expression level for each gene

Table 2: MSE (16) of the 12 MISO models for the gene prediction, for the training and testing data subsets

Predicted gene	Prediction errors		
	Cdc15 data set	*Cdc28* data set	*Alpha* data set
SIC1	0.4542	0.4059	0.7443
CLB5	0.1721	0.1769	0.4459
CDC20	0.5523	0.3661	0.6190
CLN3	0.2493	0.2473	0.1490
SW16	0.2833	0.3308	0.4983
CLN1	0.1874	0.3641	0.6653
CLN2	0.5642	0.5753	0.7346
CLB6	0.4005	0.3646	0.2523
SWI4	0.3604	0.4903	0.1218
CDC28	0.1300	0.0675	0.0576
MBP1	0.2742	0.4294	0.6993
CDC6	0.3656	0.3366	0.4235

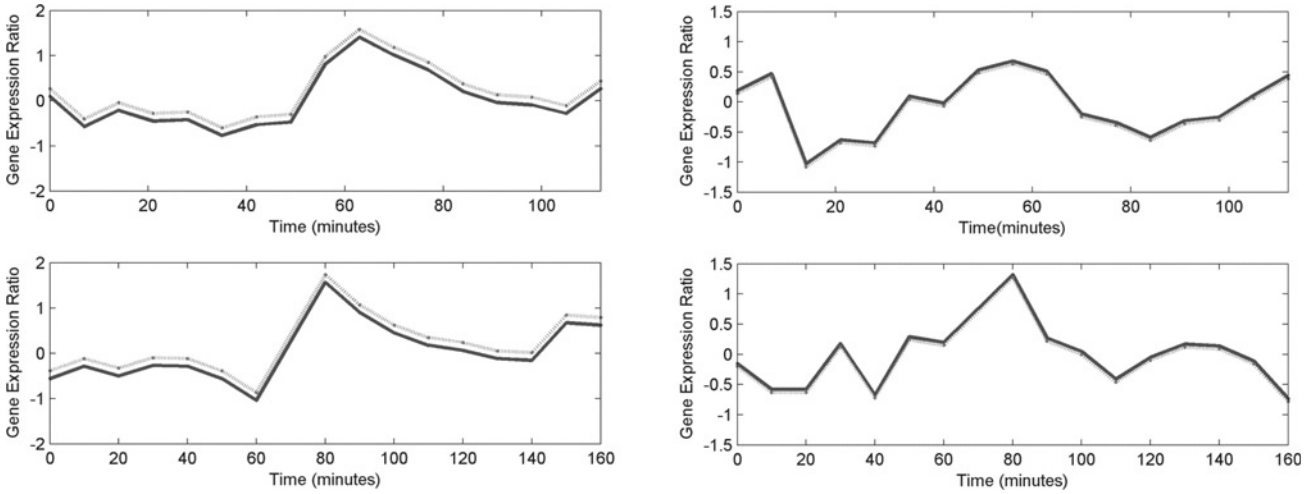

Fig. 5 *Predictions of gene expression patterns*

Left two graphs represent the predictions for gene SIC1 while the right two represent the predictions for the gene CLN3
For both genes, the upper plot corresponds to the alpha data subset while the lower plot corresponds to the *cdc28* data subset (solid lines represent actual expression while the dotted lines represent the prediction of the model)

fits to the actual values, another proof that the method manages to recover good results without overfit.

In a parallel experiment and in order to validate our approach with a second biological data set, we used the data of *E. coli.* The specific data set studies the SOS system that has a 'single input module' architecture [20], where a single transcription factor controls multiple-output operons, all with the same regulation sign (repression or activation), and with no additional inputs from other transcriptional factors. This is a basic recurring architecture in transcriptional networks [21] and characterises more than 20 different gene systems in *E. coli* [20].

Following the same methodology we have described so far in the current section, we create nine MISO NFRN systems that describe each one of the genes involved in the process, on the basis of input taken from all remaining genes. In the work of Ronen *et al.* [19], data from only the first and second microarray experiments were used. The first experiment data set was used for training and the second as a set of data for testing the parametrisation algorithm adapted by the authors. In this paper, we use the same data for training the NFRN models, while we test the accuracy on all the remaining data sets (results concerning Exp. 4 are not shown). The results of the experiment are shown in Table 4 and Fig. 6. As can be observed in Table 4, our method manages to recover better results than that of Ronen *et al.* [19] when both training and prediction data sets originated from the same experiment. Also observe the accuracy when the training and prediction data were chosen from different microarray experiments. Ronen *et al.* [19] argued that a basic limitation of the approach used in the present study was its limitation in capturing systems with multiple varying transcriptional factors like the yeast data set, while providing good results in systems of a single transcription factor, as in the SOS data. Results have proved that the approach presented in this

Table 3: MSE (16) errors of the 12 MISO models for the gene prediction (refer to the artificial data set)

Predicted gene	Prediction errors		
	1st data set	2nd data set	3rd data set
Gene 1	0.3886	0.5383	0.5662
Gene 2	0.6921	0.4806	0.7095
Gene 3	0.4615	0.2891	0.7121
Gene 4	0.3278	0.4038	0.2035
Gene 5	0.2769	0.2586	0.2728
Gene 6	0.2277	0.2131	0.7032
Gene 7	0.5851	0.4080	0.3758
Gene 8	0.3584	0.4904	0.4160
Gene 9	0.6031	0.5472	0.3624
Gene 10	0.2090	0.1099	0.1168
Gene 11	0.0589	0.1247	0.6840
Gene 12	0.2534	0.3890	0.2879

Columns contain the errors resulting from the 3-fold cross validation (2/3 of the data is used as the training set and the remaining 1/3 as the test set)

Table 4: E^1 and $E^{Previous}$ are the mean errors obtained when both training and testing data were part of the first microarray experiment

Gene	E^1	E^2	$E^{Previous}$	Function
uvrA	0.090	0.115	0.14	Nucleotide escision repair
lexA	0.084	0.105	0.10	Transcriptional repressor
recA	0.100	0.120	0.12	Mediates LexA autocleavage blocks replication forks
umuD	0.085	0.200	0.21	Mutagenesis repair
polB	0.079	0.302	0.31	Translesion DNA synthesis, replication fork recovery
ruvA	0.204	0.201	0.22	Double-strand break repair
uvrD	0.172	0.195	0.20	Nucleotide escision repair, recombinational repair
uvrY	0.16	0.420	0.45	SOS operon of unknown function

E^2 describes the errors when we used data for testing that were part of the second microarray experiment where our method also managed to get adequate results. In this table, we present the accuracy of our system, in terms of E^1 and E^2, as well as for the parametrisation algorithm used by Ronen *et al.* [19] $E^{Previous}$, using the mean error (17) metric criterion on the *E. coli* data set

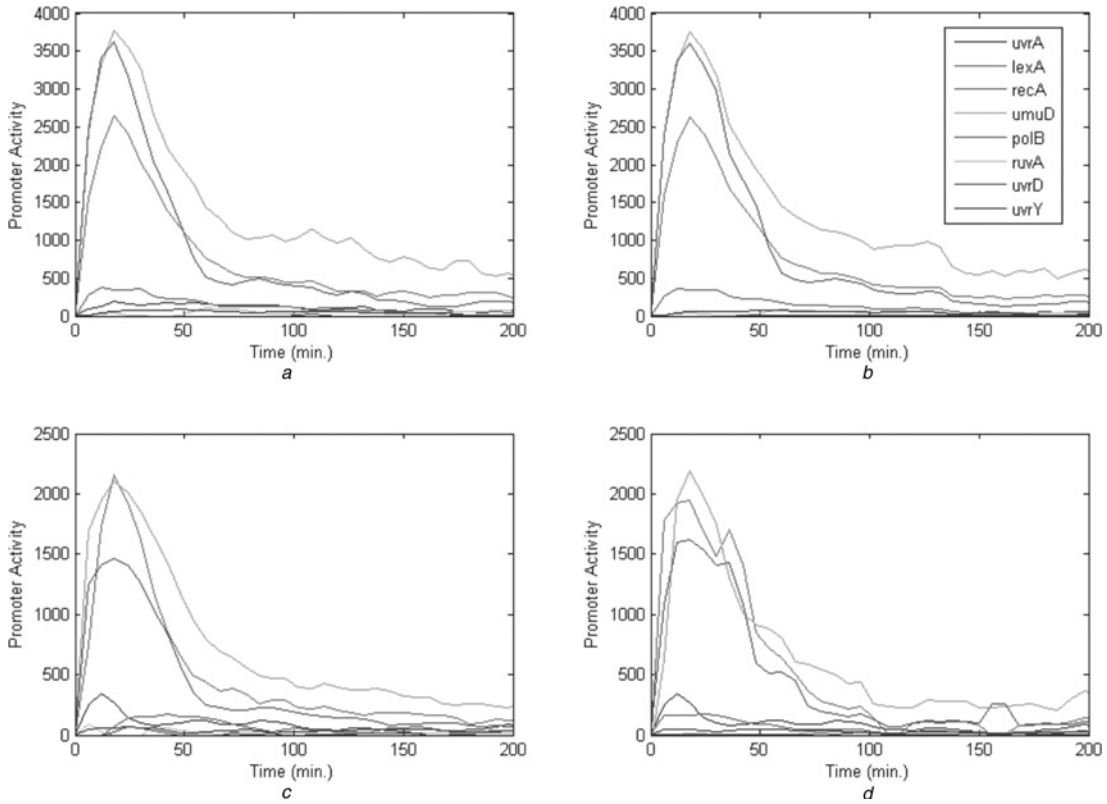

Fig. 6 *Comparison of experimental [graphs (a) and (c)] gene expression ratios from the second and third experimental data set of E. coli, respectively, and the NFRN predicted time series [graphs (b) and (d)]*

In both cases of prediction, the data for the training of the NFRN models was derived from the first microarray experiment

study manages to produce accurate and realistic results for both gene architectural systems.

In the final step of our analysis, we dealt with the data sets concerning the yeast. The rule base describing the gene interactions were derived from the 12 NFRN models used to build a gene network structure (Fig. 7) in the form of a graph in which each node represents a gene and the presence of an edge between two nodes indicating an interaction between the connected genes (either activation – represented by arrows, or repression – represented by closed circles). From a total of 36 rules derived by the 12 network models, 19 were kept for being consistent throughout the models.

Biologically, accurate interactions have been successfully extracted using our method, such as the positive regulation of CLB6 by MBP1, the inhibition of CLB5 by CDC20, the positive regulation of CLN1 by CLN2 as well as the negative regulation of the same gene by CDC20. CLN3 activates by phosphorylation the transcription factor group SBF (formed by SWI4 and SWI6), which in turn activates the cyclin CLN2. Other successfully found interactions include those between CLB5 and CLB6, while MBP1 (part of the transcription factor group MBF) was found to activate the cyclin CLB6. These are relations that besides their biological confirmation have been determined by other approaches [22, 23]. But there are cases like the positive regulation of SWI4 by MBP1 that both the supervised learning analysis of Soinov *et al.* [22], as well as the linear fuzzy approach of Sokhansanj *et al.* [23] failed to extract. It should be noted that the last relation described was successfully extracted despite the fact that the MBP1 transcription varied within a small range, which could have led to a potential error produced by the model.

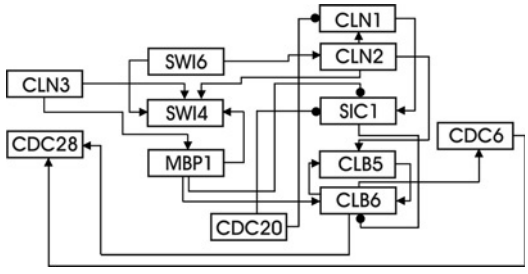

Fig. 7 *Graph of the derived gene network interactions*

Figure provides a schematic of the rules extracted using all the models in our application

4 Discussion

Employing the NFRN structure, we have found relations in the form of fuzzy rules, among the 12 yeast genes we analysed. We have used the yeast data set to experiment in the reconstruction of gene regulatory networks for two main reasons. The first is that the number of genes in this data set is larger than in the SOS data set, and those genes share multiple varying transcriptional factors providing the ability to test our method in a more complicated environment. The second motive was that much more work in terms of bioinformatics techniques has been done on the yeast data set allowing for a better judgement on the acquired results of the proposed approach.

We have also checked our method against a Bayesian network-based technique. A study of two different versions of Bayesian networks: the dynamic Bayesian networks (DBN) and the dynamic differential Bayesian nets (DDBN) are presented in the work of Sugimoto and Iba [24] and Kim *et al.* [25].

Table 5: Relations among genes that have been successfully detected by our method and failed to be extracted by DBN or DDBN

Relations among genes	Prediction errors		
	Our method	DBN	DDBN
MBP1 → CLB6	√	—	—
CDC20 → CLB5	√	—	—
CDC20 → CLN1	√	—	—
CLN2 → SWI6	√	—	√
CLB6 → CLB5	√	√	—

These methods cannot find regulations among genes in terms of positive or negative regulation, but only determine relations among genes. Our method was successful in determining over 90% of the correct relations found by each one of the methods. In Table 5, we present relations that were successfully found by our approach and failed to be identified by either or both of the two Bayesian methods.

There are certain cases where accurate biological relations were absent from the network model of Fig. 7 like the positive regulation SWI6 by CDC28 or the positive regulation of SIC1 by CDC28, or cases of relations that were retrieved but were inconsistent with biological knowledge, like the positive regulation of CDC28 by CDC6, instead of reverse regulatory interaction that is the correct one (positive regulation of CDC6 by CDC28).

Determining the accurate, in terms of biological knowledge, interactions among genes is an issue that must be answered by querying and analysing additional data sets from possibly new experiments. Those new data will allow the refinement of fit errors, to produce new and more pragmatic gene networks. However, the results presented in this paper prove the accuracy and efficiency of our method in successfully capturing the interactions among the considered genes, despite the noise inherited from the microarray hybridisation and the small amount of samples in the data. In view of this fact, our approach could be used to generate new hypotheses that will be very useful in designing new experiments.

5 Conclusion

The paper presents a method to extract causal interaction relationships among genes and to tackle the inverse problem of gene regulatory network reconstruction from microarray expression data. Towards this goal, we have presented a recurrent neural fuzzy architecture that is able to achieve the task in a fast and comprehensive manner. The self-organising structure of the method helps in retrieving the optimal number of relationships underlying the data while its recurrent part is able to take into account dynamic aspects of gene regulation. To our knowledge, it is the first application of a recurrent neural fuzzy approach to the problem of gene regulatory network reconstruction.

Although our approach follows the current ideology – regarding gene network reconstruction – focusing attention on specific subsystems that are easier to analyse and feasible in terms of collecting the necessary experimental data, it is able to supply starting points for deciphering the multiple complex biological systems. The inferred information provides biological insights that can be used by biologists to design and interpret further experiments.

The results prove the solid performance of a hybrid neuro-fuzzy approach, which is able to extract from a certain data set more biological meaningful relations than other computational approaches. The vast majority of the causal relationships among genes are in complete accordance with the known biological interactions in the yeast cell cycle. Although in the present study we limit the number of considered genes in order to point out and emphasise the validation of the methodology proposed, the method could be easily modified to process larger sets, eventually even entire genome-scale expression data, from which subsets of genes involved in specific regulatory processes might be identified through suitable gene selection methods.

6 Acknowledgments

The work conducted in our laboratory was supported by a grant from the General Secretariat of Research and Technology, Ministry of Development of Greece (013/PENED03) to A.B.

7 References

1 DeRisi, J.L., Iyer, V.R., and Brown, P.O.: 'Exploring the metabolic and genetic control of gene expression on a genomic scale', *Science*, 1997, **278**, pp. 680–686
2 Mavroudi, S., Papadimitriou, S., and Bezerianos, A.: 'Gene expression data analysis with a dynamically extended self-organizing map that exploits class information', *Bioinformatics*, 2002, **18**, pp. 1446–1453
3 Eisen, M.B., Spellman, P.T., Brown, P.O., and Botstein, D.: 'Cluster analysis and display of genome-wide expression patterns', *Proc. Natl. Acad. Sci.*, 1998, **95**, pp. 14863–14868
4 Golub, T.R., Slonim, D.K., Tamayo, P., *et al.*: 'Molecular classification of cancer: class discovery and class prediction by gene expression monitoring', *Science*, 1999, **286**, pp. 531–537
5 Liang, S., Fuhrman, S., and Somogyi, R.: 'REVEAL, a general reverse engineering algorithm for inference of genetic network architectures'. Pacific Symp. on Biocomputing, 2000, pp. 18–29
6 Tegner, J., Yeung, M.K., Hasty, J., and Collins, J.J.: 'Reverse engineering gene networks: integrating genetic perturbations with dynamical modeling', *Proc. Natl. Acad. Sci.*, 2003, **100**, pp. 5944–5949
7 Friedman, N., Linial, M., Nachman, I., and Pe'er, D.: 'Using Bayesian networks to analyze expression data', *J. Comp. Biol.*, 2000, **7**, pp. 601–620
8 Ling, F., Long, T., Lu, Y., Ouyang, Q., and Tang, C.: 'The yeast cell-cycle network is robustly designed', *Proc. Natl. Acad. Sci.*, 2004, **101**, pp. 4781–4786
9 Mastrocostas, P.A., and Theocharis, J.B.: 'A recurrent fuzzy-neural model for dynamic system identification', *IEEE Trans. Syst., Man Cybern.*, 2002, **32**, pp. 176–190
10 Juang, C.F.: 'A TSK-type recurrent fuzzy network for dynamic systems processing by neural network and genetic algorithms', *IEEE Trans. Fuzzy Syst.*, 2002, **10**, (2), pp. 155–170
11 Juang, C.-F., and Lin, C.-T.: 'A recurrent self-organizing neural fuzzy inference network', *IEEE Trans. Neural Netw.*, 1999, **10**, pp. 828–845
12 Zadeh, L.A.: 'Fuzzy sets', *Inform. Control*, 1965, **8**, pp. 338–353
13 Lin, C.T., and Lee, C.S.: 'Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems', *IEEE Trans. Fuzzy Syst.*, 1994, **2**, pp. 46–63
14 Lee, C.H., and Teng, C.C.: 'Identification and control of dynamic systems using recurrent fuzzy neural networks', *IEEE Trans. Fuzzy Syst.*, 2000, **8**, pp. 349–366
15 Williams, R.J., and Zipser, D.: 'A learning algorithm for continually running recurrent neural networks', *Neural Comp.*, 1989, **1**, (2), pp. 270–280
16 Werbos, P.: 'Beyond regression: new tools for prediction and analysis in the behaviour sciences'. PhD Dissertation, Harvard University, Cambridge, MA, USA, August 1974
17 Spellman, P.T., Sherlock, G., Zhang, M.Q., Iver, V.R., Anders, K., Eisen, M.B., Brown, P.O., Botstein, D., and Futcher, B.: 'Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization', *Mol. Biol. Cell*, **9**, pp. 3273–3297
18 Oba, S., Sato, M., Takemasa, I., *et al.*: 'A Bayesian missing value estimation method for gene expression profile data', *Bioinformatics*, 2003, **19**, pp. 2088–2096
19 Ronen, M., Rosenberg, R., Shraiman, B.I., and Allon, U.: 'Assigning number to the arrows: parameterizing a gene regulation network by using accurate expression kinetics', *Proc. Natl. Acad. Sci.*, 2002, **99**, pp. 10555–10560

20 Shen-Orr, S.S., Milo, R., Mangan, S., and Alon, U.: 'Network motifs in the transcriptional regulation network of *Escherichia coli*', *Nat. Genet.*, 2002, **31**, pp. 64–68
21 Neidhardt, F.C., and Savageau, M.A.: 'Regulation beyond the operon' in Neidhardt, F.C. (Ed.): 'Escherichia coli and Salmonella: cellular and molecular biology' (American Society of Microbiology, Washington DC, 1996, 2nd edn), pp. 1310–1324
22 Soinov, L., Krestyaninova, M., and Brazma, A.: 'Towards reconstruction of gene networks from expression data by supervised learning', *Genome Biol.*, 2003, **4**, pp. R6.1–R6.10
23 Sokhansanj, B., Fitch, P., Quong, J., and Quong, A.: 'Linear fuzzy gene network models obtained from microarray data by exhaustive search', *BMC Bioinform.*, 2004, **5**, pp. 1–12
24 Sugimoto, N., and Iba, H.: 'Inference of gene regulatory networks by means of dynamic differential Bayesian networks and nonparametric regression', *Genome Inform.*, 2004, **15**, (2), pp. 121–130
25 Kim, S., Imoto, S., and Miyano, S.: 'Dynamic Bayesian network and nonparametric regression for nonlinear modelling of gene networks from time series gene expression data', *Biosystems*, 2004, **75**, pp. 57–65