

Searching for Limited Connectivity in Genetic Network Models

E.P. van Someren[§], L.F.A. Wessels^{§,†}, M.J.T. Reinders[§] and E. Backer[§]

[§]Information and Communication Theory Group, [†]Control Laboratory

Delft University of Technology
Mekelweg 4, Delft, The Netherlands
E.P.vanSomeren@its.tudelft.nl

ABSTRACT

The inference of regulatory interactions between genes from time-course micro-array data is one of the most challenging tasks in the field of functional genomics. The multitude of genes that can now be measured using micro-array technology requires analysis tools that can easily scale-up with respect to the number of genes. This scalability is especially important when inferring genetic interactions, because this task is complicated by the combinatorial nature of gene interaction and because the high cost of micro-array measurements still severely limits the number of measured time-points. Because of this limitation of the data, it is essential to incorporate as much additional information as possible. This can be achieved by applying constraints based on general biological knowledge and by including specific knowledge about known interactions. In this paper we employ the fact that genetic networks are believed to exhibit limited connectivity. We propose a general approach in which we separate the task of finding the structure of the networks from the task of finding the best parameters of the model, given the structure. The second task can be solved efficiently for most models, but the first task amounts to a search problem which requires the choice of a suitable evaluation function and search strategy. Experimental investigations determined that the best evaluation function is simply the mean squared error on the training data. Through further extensive experimental investigation of several search strategies, it was found that the best search strategy is based on an approach of greedily increasing the number of connections. The strength of the proposed approach lies in the fact that it can be employed to *all* genetic network models and allows genetic network models to scale up to a large number of genes.

1. INTRODUCTION

Both in biology and in medicine a lot of research is directed towards discovering, understanding and/or controlling the outcome of some particular biological pathway. A multitude of examples exist, where the manipulation of a key enzyme in such a pathway did not lead to the desired/expected effect, because the intended effect was compensated for by the genetic regulation of the enzyme levels. Such examples illustrate how important it is to take into account the effect caused by genetic regulation in order to truly understand and control the outcome of many of the biological processes that take place in the cell. Gene expression levels them-

selves are regulated by the combined action of multiple gene products [12]. Now, one of the truly challenging questions in systems biology is whether it is possible to model these genetic interactions as a network of interacting elements and whether these interactions can be effectively learned from measured time-course gene-expression data. The goal of genetic network modeling is, thus, to improve the discovery of genetic pathways by generating hypotheses of likely interactions between genes, which then need to be further validated experimentally.

In general with genetic network modeling, the genetic interactions are represented by parameters in a parametric network model. A genetic network solution is then obtained by finding a suitable set of parameters such that the measured data can be adequately predicted by the model. However, because of current micro-array technology, time-course gene expression data typically contains the expression of *many genes* (thousands) measured over a *limited number of consecutive time-points* (generally, less than twenty). In addition, the measurements are corrupted by a *substantial amount of uncharacterized measurement noise*. The large number of elements, the limited number of noisy observations together with the combinatorial nature of the problem makes it particularly difficult for a data-driven approach to reliably extract the large number of complex interactions. This consequence of the, so called, "dimensionality problem" (many genes, few time-points) can, generally, be relieved by *reducing the complexity* of the employed network model and by *constraining the parameters* based on other available knowledge. Furthermore, the inference-algorithm that is used to learn the parameters, must be *robust against noise* and easily *scale-up with the number of genes*, both in terms of performance as well as in terms of the necessary computational costs.

In this paper, a discrete-time linear genetic network model is chosen, because it contains relatively few parameters that are easy to interpret and which can be learned very fast and robustly using standard linear algebra techniques. However, the techniques and observations presented in this paper are not necessarily limited to this linear model. In previous work [18], we have successfully improved the performance of genetic network models by applying constraints on the robustness of genetic networks. Because gene expression is known to be regulated by a limited number of gene products [3], we propose, in this paper, to limit the connectivity of genetic network models and to search for the most likely connections. By constraining the number of connections,

the resulting models will not only resemble their biological counterparts more, but the "dimensionality problem" will be relieved and the model parameters will be learned much more reliably.

One extreme case of models that restrict the number of connections are *pair-wise models* [2, 4, 23]. These models determine possible relationships by comparing gene expression profiles in a pair-wise fashion only, which makes the inference fast and untroubled by the dimensionality problem. A disadvantage of pair-wise models is, however, that their results are based on an effectively singly-connected network. We would like to use models that take into account the fact that gene expression is caused by the *combined action* of multiple gene products [12], but still use *as few connections as possible*. Other approaches have suggested a specific search strategy for a specific model on relatively small networks. The REVEAL algorithm proposed by Liang [14] employs a forward¹ breadth-first search procedure to find the structure of Boolean models of 50 genes. The approach employed by Weaver [20] follows a greedy backward² search to find the parameters of a quasi-linear network³ of 50 genes and Wahde [19] utilizes a genetic algorithm to get a distribution of the parameters of a non-linear model of 15 genes and determines the structure in a backward fashion by greedily clamping the least significant parameters to zero. Friedman [10], on the other hand, has shown that a Bayesian Network can be learned using a greedy hill-climbing method on 800 genes that were believed to be cell-cycle regulated, but this model did not allow cyclic networks nor dynamic behavior.

Rather than presenting a specific strategy for a specific model, we propose a more *general* approach that is *scalable* to large networks and evaluate many different search strategies. Our approach is based on separating the task of finding the structure of the network from the task of finding the actual parameter values for a given structure. The task of finding the structure can be defined as a general search problem, which can be solved efficiently and independently of the chosen model once a proper evaluation function⁴ and search strategy have been chosen. The task of determining the best parameters can be solved efficiently (accurately and rapidly) by existing algorithms for many of the available models. Consequently our approach is 1) *general*, i.e. it can be applied to all models that have a reasonably fast way of determining the weights for a given structure and is 2) *computationally fast*, making it scalable to large networks (hundreds of genes). Furthermore, we have compared *several different search strategies* and investigated *different evaluation functions* and show that this approach can find the right connections back reasonably well under realistic conditions, i.e. under noisy conditions and for networks of up to at least 1000 genes.

The remainder of this paper is structured as follows. First the considerations for choosing a proper model and inference algorithm are further addressed and the problem of finding the network structure is described as a search problem. Subsequently, based on extensive experimental investigations, some insights are given concerning the best evaluation func-

¹increasing the number of connections

²decreasing the number of connections

³a neural network without hidden layer or, equivalently, a linear network augmented with a sigmoid

⁴a function that determines how "good" a solution is

tion and search strategy and finally some considerations for future work are given.

2. CHOOSING THE MODEL

Thus far, a large number of different modeling approaches have been proposed, among which are Boolean networks [14], Bayesian networks [10, 11], (Quasi)-Linear networks [15, 9, 20, 19] and Differential Equations [5]. The complexity of these models varies with 1) how gene expression levels are represented, i.e. with discrete or continuous values; 2) how dynamic behavior is modeled, i.e. discrete- or continuous-time systems and 3) what kind of relationships between genes are allowed, i.e. lattice, linear or nonlinear functions. An extensive comparison between the different models is beyond the scope of this paper, more can be found in [7, 6]. In [22, 17], we presented comparative studies with experimental investigations. Here, we will suffice by summarizing some properties that reflect the complexity of these models (See Table 1).

Because it is important to be able to learn large networks, i.e. with more than 100 genes, the model should be as simple as possible in order to come up with reliable results. One way to achieve this is by choosing a model that represents gene expression using a limited number of discrete levels rather than by continuous values. However, we prefer a continuous representation because the data is also represented by continuous values and this way we avoid reducing the already limited information of the data-set by an, otherwise necessary, discretization method.

Another model choice is whether to represent dynamic behavior with a discrete- or continuous-time model. True gene expression levels change continuously over time. The measurements, however, are taken at discrete points in time. We prefer a discrete-time representation because it represents the data more effectively and avoids discretization of a continuous model. To limit the number of parameters we choose to take only the previous time-step into account, thereby employing the central dogma that the state of the cell is completely determined by the state of all gene expression levels at a certain point in time. For now, we assume that this limitation can only be lifted once more data becomes available.

To sufficiently simplify the model we allow only linear relationships for our continuous-level discrete-time model. This simplifies the model (only N^2 parameters for an N -gene network), makes the parameters (linear weights) interpretable and allows the parameters to be learned fast and robustly using standard linear algebra techniques.

3. CHOOSING THE INFERENCE METHOD

For a given model, the parameters need to be inferred from the measured gene expression data. A good inference method must, in addition, be able to restrict the number of connections, be able to cope with large networks and be robust against measurement noise.

A different way to resolve the dimensionality problem, apart from using a simple model, is to constrain the parameters of the model by employing other available knowledge. For example, genetic networks are assumed to be sparsely connected [3], robust against noise [9], redundant [21], and stable. In addition, the expression levels of genes are assumed to behave *smoothly* over time [9]. Other knowledge con-

Table 1: Complexity properties of some proposed models. Columns indicate 1) the proposed model; 2) whether gene expression is represented with discrete (2,3) or continuous (∞) values; 3) whether dynamics are represented by a discrete (discr.) or continuous (cont.) system or not at all (-); and if so, 4) how many previous time-points are taken into account; 5) which type of relations are allowed and 6) how many connections are searched for.

Model Reference	Expression Level	Dynamics		Relations	Connections
		type	memory		
Liang98a [14]	2	discr.	1	Boolean	$< N$
D’Haeseleer98a [8]	∞	discr.	1	linear	N
Someren00b [16, 18]	∞	discr.	1	linear	$< N$
Weaver99a [20]	∞	discr.	1	linear+sigmoid	$< N$
Arkin97a [2]	∞	discr.	N	correlation	1
Chen99b [4]	∞	discr.	N	peaks	1
Wahde99a [19]	∞	cont.	1	linear+sigmoid	$< N$
Friedman [10]	3	-	-	lattice	$< N$
"	∞	-	-	linear	$< N$

sists of specific information about known relationships between particular genes that has been acquired through other means.

Here, we focus on how we can exploit the fact that genetic networks have limited connectivity. In [3], Arnone and Davidson conclude from studying all known cis-regulatory regions that gene expression is being influenced on average by between 4-8 different gene products. This provides an indication that the expected connectivity of many genes is indeed limited. Jeong [13] indicates that the number of incoming as well as outgoing connections follows a power-law distribution and that thus there is still a large number of connections that can correspond to one gene. Incorporating limited connectivity implies that we are looking for *sparse* linear networks.

4. LINEAR GENETIC NETWORK MODEL

Practically, all the currently proposed models are based on a parameterized model, where the parameters need to be inferred from the gene expression data. A common element in these models is that they represent the actual gene-gene interactions with a set of parameters to which the following interpretation is given. In all currently employed models, a genetic interaction is represented with a single parameter, w_{ij} , which represents how gene j controls gene i . The interaction parameter, $w_{i,j} \in \mathbb{R}$, represents a controlling action of gene j on gene i , whether it is activating ($w_{i,j} > 0$) or inhibiting ($w_{i,j} < 0$), as well as the strength ($|w_{i,j}|$) of the relation. The complete matrix of interactions, \mathbf{W} , is called the gene regulation matrix. In the case of a linear model it is assumed that the gene expression level of each gene is the result of a weighted sum of all other gene expression levels at the previous time-point:

$$g_i(t+1) = \sum_{j=1}^N w_{i,j} \cdot g_j(t) \quad i, j \in \mathbb{N}_N^+, t \in \mathbb{N}_{T-1}^+ \quad (1)$$

Here, $g_i(t)$ represents the gene expression level of gene i (out of N genes) at time t (out of T time-points) and $\mathbb{N}_K^+ = [1, K]$. This simple linear model, can be found at the core of the continuous-level genetic network models. The existence or absence of a relationship is represented by a non-zero ($w_{i,j} \neq 0$) or zero ($w_{i,j} = 0$) parameter value.

For the purpose of this paper, we want to decouple the struc-

ture of the networks from the strength of the relationships. Therefore, we depart from the classic representation by making explicit whether or not a relationship exists. A distinction is made between the strength of the relationship, $w_{i,j}$, and the existence of a relationship by introducing an additional variable, $c_{i,j} \in \{0, 1\}$:

$$g_i(t+1) = \sum_{j \in \mathcal{C}_i} w_{i,j} \cdot g_j(t) \quad i, j \in \mathbb{N}_N^+, t \in \mathbb{N}_{T-1}^+,$$

$$\mathcal{C}_i = \{ j \mid j \in \mathbb{N}_N^+, c_{i,j} = 1 \} \quad (2)$$

Here, the connection parameter, $c_{i,j} \in \{0, 1\}$, represents the existence ($c_{i,j} = 1$) or absence ($c_{i,j} = 0$) of a controlling action of gene j on gene i . The interaction parameter, $w_{i,j} \in \mathbf{W}$, now represents the strength ($|w_{i,j}|$) of the controlling action of gene j on gene i and whether it is activating ($w_{i,j} > 0$) or inhibiting ($w_{i,j} < 0$). Consequently, the connection matrix, $\mathbf{C} = [c_{i,j} \mid i, j \in \mathbb{N}_N^+]$, now represents the structure of the genetic network. Reducing the number of connections implies that we are looking for network structures, \mathbf{C} , that have only a few non-zero weights on each row.

5. THE SEARCH FOR CONNECTIVITY

The problem of finding a network of genetic interactions can be divided in two ways. Firstly, the prediction of each gene’s expression value depends only on the expression values of all genes at previous time-points. As this input is the same for all genes, the parameters corresponding to the prediction of each gene can be learned independently of the parameters of the other genes. Therefore, for N genes the problem of finding a network of N -to- N possible interactions reduces to finding N separate networks of N -to-one interactions (see Figure 1). A disadvantage of this simplification⁵ is that the consequences of small prediction-errors made in previous time-steps are ignored during the learning-process. We note here that it is not our goal to construct a system that can perfectly reproduce the real genetic network, but rather to learn a simplified model that still reflects information about the true connectivity.

Secondly, the task of finding the parameters corresponding to the prediction of one gene can be further divided in a

⁵This simplification is also commonly used in system theory

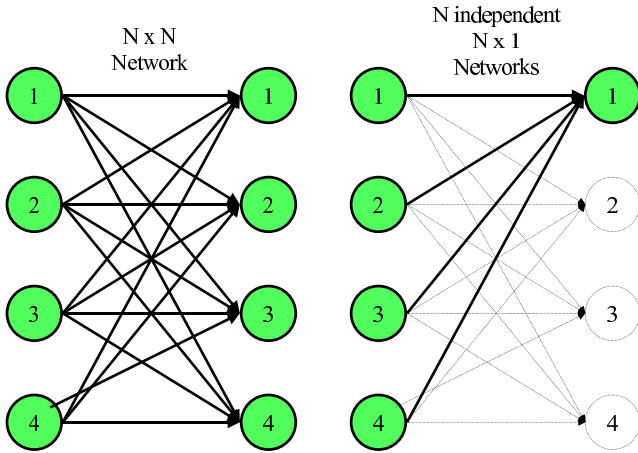


Figure 1: The prediction of each gene is independent.

step that determines the structure⁶ of the network (which connection parameters, $c_{i,j}$ are non-zero), followed by a step that determines the actual values of the weights, $w_{i,j}$ that correspond to the given structure and the data. In general, the structure cannot be determined directly from the data. Instead, an iterative procedure is followed which 1) generates a hypothesis of the structure, 2) determines the weights based on the structure, 3) determines the predicted signals, 4) determines a fitness⁷ of that model solution, and 5) based on the fitness, stops the search or starts the cycle again at Step 1 with an adjusted hypothesis. This search cycle is illustrated in Figure 2. Two choices are essential to complete this cycle, that is a *fitness function* needs to be defined such that each possible solution can be assigned a fitness-value that represents how well it matches the original (unknown) model and a *search strategy* needs to be defined such that the fittest solution is found after investigating as few solutions as possible.

5.1 Fitness Function

The fitness function needs to distinguish good solutions from bad solutions. An ideal fitness function assigns higher fitness values to solutions that are closer to the perfect solution and assigns the highest possible fitness value to the perfect solution. In practice, the perfect solution is not known and thus an ideal fitness function cannot be found, but a function needs to be chosen instead that approximates this behavior as closely as possible. When inferring genetic network models only the measured data-set is available and, therefore, the assumption is used that a good genetic network model should generate data close to the measured data-set. Consequently, one possible choice of the fitness function is based on taking the mean squared error on the training data, i.e. between the predicted and the measured data, where the same data is used during learning and prediction. Be-

⁶In the remainder of this paper we will sometimes refer to the structure of the network even if it concerns only a part of it, e.g. the part corresponding to the prediction of one gene.

⁷With fitness we denote a score for the evaluation of a solution

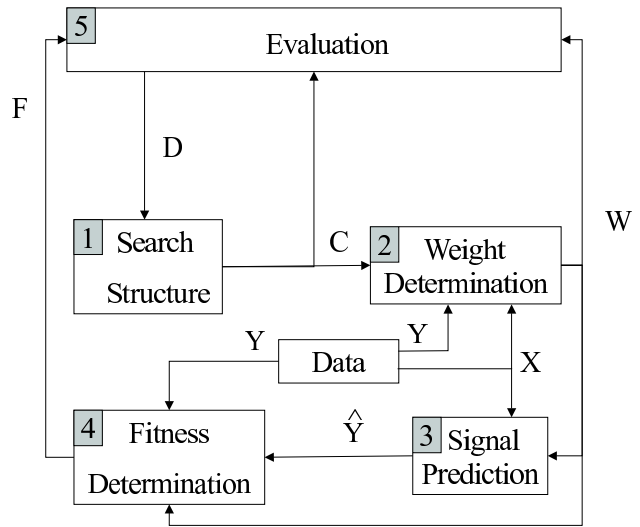


Figure 2: A block diagram of the steps taken in the search cycle.

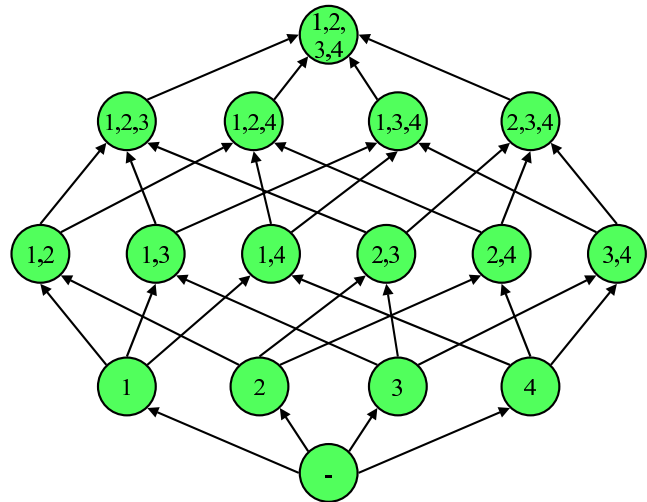


Figure 3: The hierarchical structure of the search space in the case of only 4 genes.

cause the averaging effect of the mean-operator, this function might favor models that disregard single peaks in the data. An alternative fitness function is based on the maximum squared error on the training data. This function is more sensitive to noise, but gives only high scores to solution that fit all points in the data well. An important consideration is that by increasing the number of connections (thus making models more complex) the data-fit on the training data will always improve and a fully connected network will give a perfect data-fit⁸. However, we are interested in finding sparsely connected networks and we want the models to be robust against noise, therefore, over-fitting of the data⁹ needs to be avoided. Consequently, a third alternative for the fitness function, that is commonly used to avoid over-fitting, is based on a leave-one-out cross-validation. Such a fitness function will favor models that have good generalizing behavior and thus performs well on unseen data. However, this function takes a much longer time to be determined (a property which might become undesirable as this function needs to be executed many times, i.e. for each considered solution). The three fitness functions are:

- **Mean Squared Error Fitness**

$$F^{Mean}(\mathbf{Y}_i, \hat{\mathbf{Y}}_i) = \frac{1}{1 + \frac{1}{T-1} \sum_{t=1}^{T-1} (y_i(t) - \hat{y}_i(t))^2} \quad (3)$$

- **Maximum Squared Error Fitness**

$$F^{Max}(\mathbf{Y}_i, \hat{\mathbf{Y}}_i) = \frac{1}{1 + \max_t [(y_i(t) - \hat{y}_i(t))^2]} \quad (4)$$

- **Leave-One-Out Error Fitness**

$$F^{Loo}(\mathbf{Y}_i, \hat{\mathbf{Y}}_i) = \frac{1}{1 + \frac{1}{T-1} \sum_{t=1}^{T-1} (y_i(t) - \hat{y}'_i(t))^2} \quad (5)$$

Here, $\hat{y}_i(t) = \mathbf{W}_i \cdot \mathbf{X}^t$ is the t -th predicted output expression value of gene i and \mathbf{X}^t is the t -th input state and the solution $\mathbf{W}_i = \mathcal{I}(\mathbf{X}, \mathbf{Y})$ was learned using all measured in- and outputs. Alternatively, $\hat{y}'_i(t) = \mathbf{W}'_i \cdot \mathbf{X}^t$ is again the t -th predicted output expression value of gene i , but now each t -th sub-solution $\mathbf{W}'_i = \mathcal{I}(\mathbf{X}/\mathbf{X}^t, \mathbf{Y}/\mathbf{Y}^t)$ was learned using all measured in- and outputs, *except for the t -th input-output pair*.

To determine the best fitness function an extensive study was done on which one performs best. All experiments in this paper were done using GENLAB, a gene expression analysis toolbox developed by the authors and available at <http://www.genlab.tudelft.nl>. First, a multitude of gene expression data-sets was generated but each data-set was generated using the following procedure: 1) a random, stable and sparse gene regulation matrix of a given number of genes and connectivity was chosen; 2) four different random initial gene expression states were chosen; and for each of these initial states, 3) four additional consecutive gene-expression

⁸Due to the dimensionality problem a perfect data-fit can already be achieved for a less than fully-connected network.

⁹In general, when a model is over-fitting it means that it fits the training data too perfectly and thus it fits the noise on top of the signal. In such a case the model will have a low performance on new (unseen) data and the weights do not really reflect the underlying model.

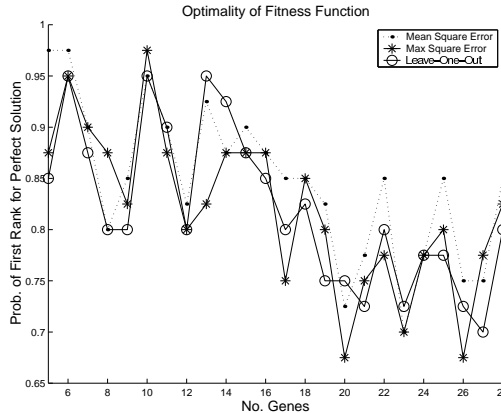


Figure 4: Performance of each fitness function in terms of how many times it ranks the perfect solution as the best.

states were generated using the chosen gene regulation matrix. Thus, each data-set consists of four sub-datasets of 5 time-points each. Consequently, each data-set consists of 20 different time-points, from which 16 input-output pairs can be used for learning. Using this procedure, we generated 40 repeats of networks with numbers of genes varying from 5 to 30, but the connectivity was kept at four. Thus in total (40 × 26) different gene regulation matrices with corresponding data-sets were generated. For each data-set, one of the genes was chosen randomly to be predicted.

Second, an exhaustive search of possible network structures was performed on each data-set as follows: Each structure that contains combinations of at most 4 connections was considered a possible solution and for each solution a proper set of weights was learned and the corresponding predictions were generated¹⁰. All three fitness functions were applied to these generated solutions making it possible to rank the solutions based on the results of either of the three fitness functions. Among all solutions one exactly matches the connections of the original gene regulation matrix and thus this perfect solution gets ranked by each of the three fitness functions. In Figure 4, the probability that the perfect solution gets the best rank is plotted for different numbers of genes. An ideal fitness function would always rank the perfect solution as the best, i.e. the probability should be one. From Figure 4 we observe that the probability of getting the perfect solution ranked best, decreases with increasing number of genes as expected. Furthermore, we can conclude that all three fitness functions perform almost equally well. The performance of the Leave-One-Out Fitness function is disappointing as we would have expected it to be better than the other fitness functions, furthermore because this function takes roughly a factor of 100 times more computational time it is not the preferred choice. The mean squared error fitness function scores slightly better than the other two and is employed as the fitness function in the rest of this paper.

¹⁰Because the number of possible structures increases very rapidly for increasing numbers of genes, the network sizes were limited to at most 30 genes for computational reasons.

5.2 Search Strategy

Given an N -to-one network, the search space of structures corresponds exactly to the search space belonging to the problem of drawing K elements out of N objects without replacement, irrespective of order and for unknown $K \leq N$ (See Figure 3). Obviously, this search space becomes enormously large for large N , and the problem of finding the right solutions is known to be NP-hard, in fact, the search space can become as large as:

$$N_{sol} = \sum_{k=1}^N \binom{N}{k} \quad (6)$$

It is evident that in order to find the right structure of large genetic networks in reasonable time the number of solutions that may be evaluated must be much lower than the total number of possible solutions. The principle choices that can be made for any search problem are the *starting point*, the *direction* of search and the *stop-criterium*. Each of the choices can greatly impact on the number of steps necessary to reach the optimum. For genetic network modeling the most obvious choice for the starting point is at a network with no connections at all, because the number of connections is believed to be quite small. However, a fully-connected network may already contain a substantial amount of information about the true connections. Therefore, both variants are considered. The search can be performed 1) in a single direction, i.e. either increasing or decreasing the number of connections, 2) in two directions, i.e both increasing and decreasing, or 3) no general direction at all (e.g. stochastic search or genetic algorithm). Although the single direction approach will be faster, the bi-directional approach allows the possibility to recover from previous errors and local minima. The stop-criterium determines when the search is complete and is primarily of consequence for the computational costs. In this paper, we did not investigate any intelligent stop-criterium, but used a fixed number of connections as a stop-criterium instead. However, we think that investigations concerning a good stop-criterium are one of the primary considerations for future research.

In this paper, we consider the following search strategies:

- **Forw- K**

This strategy first considers all solutions with single connections, picks the best solution, removes that gene from the possible inputs and subtracts its contribution from the output. Then it considers all solutions with the remaining inputs that can predict the remainder of the output and so on, up to K connections.

- **Forw-reest- K**

Similar to the previous strategy, but the weights of the previously selected connections are re-estimated. This method first considers all solutions with single connections, picks the best solution and remembers the used gene. Then it considers all solutions with two connections that contains the previously chosen connections and so on, up to K connections

- **Forw-Top D -reest- K**

Similar to the previous strategy, but not only the best solution, but the D best solutions are expanded at each iteration. In effect, the previous strategy is the same

as "Forw-Top1-reest- K ". This method first considers all solutions with single connections, picks the best D solutions and remembers their used connections as intermediate solutions. Then for each of these D intermediate solutions, it considers all solutions with an additional connection. From all these solutions again the best D are chosen and so on, up to K connections.

- **Forw-Float- K**

A floating search strategy. This method has two modes of direction, forward and backward. It is initialized in forward mode with an initial solution of two connections generated by employing "Forw-reest-2". In forward mode, all solutions with an additional connection w.r.t the current solution are considered and the best solution among those is selected as an intermediate solution. Subsequently, all solutions generated by removing one connection from the intermediate solution are considered and the best among those is selected. If that solution is better than the current, it is chosen as the next current solution and the search is continued in backward mode. Otherwise, if it is the same as the current solution, the intermediate solution is chosen as the next current solution and the search is continued in forward mode. In backward mode, all solutions generated by removing one connection from the current solution are considered and the best among those is selected. If that solution is better than all previously found solutions with the same number of connections, that solution is set as the next current solution and the search is continued backward. Otherwise, the search is changed into the forward direction with the current solution. Furthermore, when the number of connections is two or $K + 1$, the search is always set into the forward or backward direction respectively.

- **Back- K**

This strategy takes the Moore-Penrose pseudo-inverse using all inputs and outputs. From this fully connected solution, the $N - K$ smallest weights are replaced by zeros.

- **Back-reest- K**

This strategy also takes the Moore-Penrose pseudo-inverse using all inputs and outputs. But from this fully connected solution, the smallest weight is replaced by a zero and the corresponding gene is removed from the input. Then the Moore-Penrose pseudo-inverse is taken using all remaining inputs and outputs and again the smallest weight is replaced by zero and removed from the input and so on, down to K connections.

- **Genalg-SteadyState- K**

This steady-state genetic algorithm starts with a "population" of N solutions with randomly chosen structures of K connections. Two solutions are randomly chosen from this population, based on a distribution that favors solutions that are highly ranked according to the fitness function. These two "parent"-solutions produce two new "child"-solutions that will replace the two solutions with the worst fitness in the population. The "children" are produced by taking the common connections of the "parents" and exchanging some of the different connections of the "parents". In addition one connection of the new "children" is replaced by a

random connection with a 10% chance. This procedure is done $40 \times N$ times and then the best solution is taken.

- **Genalg-Gen- K**

This (more standard) generational genetic algorithm is similar to the previous strategy, except for the fact that instead of two "children", an entire new population is created at each generation. This new population is constructed using a cross-over rate of 70%, a mutation rate of 1%, a population size of $2N$ and stopping after $7K$ generations.

- **Exhaustive- K**

This strategy just considers all solutions with K or less connections and takes the best one.

- **Reference**

This model is added as a reference model and just generates a random, stable solution with 4 connections.

6. COMPARATIVE STUDIES

Comparative studies were performed to study the relative performance of each of the different search strategies. In Figure 5, the structural power of each of the different search strategies is plotted for increasing number of genes. The structural power is defined as the percentage of connections (i.e. non-zero weights) of the original regulation matrix that was found back, i.e.

$$SP = \frac{TP}{TP + FP} \quad (7)$$

Here, TP is the number of true positives and FP is the number of false positives¹¹. For reasons of clarity, the "Forw-4" and "Back-4" models are not displayed as both resulted in lower performance w.r.t the "Forw-reest-4" and "Back-reest-4" models respectively. The most remarkable observation is that the performance of all models for increasingly larger networks drops quite significantly going from 30 to 300 genes, above which it levels out to a more-or-less steady performance. The performance at networks of thousand genes ranges from fewer than one correctly identified connection to more than two correctly identified connections for the forward- $TopD$ models. The results show that the forward approach is better than the backward approach and that the bi-directional model ("Forw-float-4") performs the same as the forward model. In experiments performed on smaller networks (results not shown), the bi-directional model did show slightly improved results w.r.t. the forward strategy. In turn, the undirected search represented by both genetic algorithms perform slightly better than the forward and floating search methods. Surprisingly, the steady-state type of genetic algorithm outperforms the more standard generational genetic algorithm for networks of 600 genes and less. Of course, the exhaustive strategy would show the best performance of all strategies, but the accompanying computational costs made it impossible to learn networks larger than 30 genes. In experiments performed on smaller networks (results not shown), the $TopD$ -forward approaches did show almost identical performance w.r.t. the exhaustive search strategy, but achieved this with

¹¹In this paper, $TP + FP = 4$ for all models, as each model returns exactly four connections.

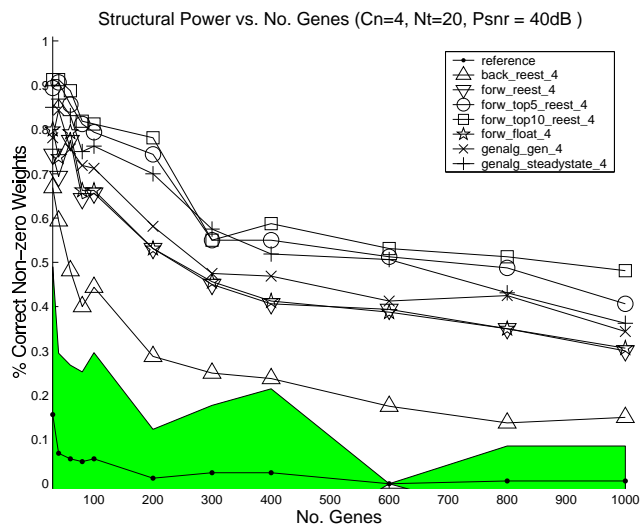


Figure 5: Performance of different search strategies for large networks in terms of how well they find back the original connections.

significantly faster computation times. Indeed, also for the larger networks, the $TopD$ approaches outperform all other models in terms of structural power, but only achieve this by evaluating a large number of possible solutions. The computational complexity, i.e. the number of solutions that are considered for each of the models, is depicted in Figure 6. First of all, we observe that the computational complexity of all approaches increases (only) linearly with the number of genes, as expected. Furthermore, when comparing Figures 5 and 6, a strong relationship can be observed between the average number of solutions a model considers and its relative performance in terms of structural power. However, the higher performance in structural power of the Forward- $TopD$ approaches w.r.t. both genetic algorithms does not follow from a larger number of considered solutions. From this we can conclude that the Forward- $TopD$ approaches are the most efficient strategies and best suited for finding structure in large genetic networks. As an indication, using a 450 Mhz Pentium, it takes approximately three and a half minutes for the Forward- $Top10$ model to find an answer for a network of thousand genes, which is quite acceptable considering the fact that this is a MATLAB implementation that was not optimized for speed at all. Furthermore, the algorithm only considered approximately 30,000 solutions out of a possible 10^{10} !

7. DISCUSSION

In this paper, we presented an approach to solve genetic network modeling by considering it as a search problem, where the most difficult part is to find the right structure of the underlying genetic network. As a result, a multitude of different algorithms that can solve this search problem are available from literature. The choice of a proper algorithm must follow from the specific properties that are associated with genetic network modeling, i.e. there are very many elements of choice, the problem is in essence combinatorial and the number of observations is severely limited and corrupted

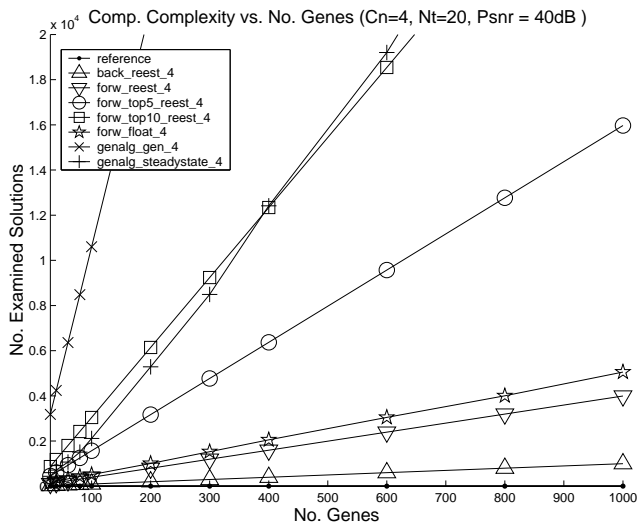


Figure 6: Performance of different search strategies for large networks in terms of the number of considered solutions.

by noise.

In this paper, we have investigated several different search strategies of which the Forward-Top D model was found to perform best. However, we believe that the current implementation provides ample room for further optimization, such as a good stop-criterion, faster implementation and better parameter setting determination. In addition, other more sophisticated search strategies need to be investigated, e.g. employing significance measures for the backward approaches or perhaps more sophisticated floating search strategies. The currently investigated evaluation (fitness) functions were all based on prediction error and all three showed similar but good performance, however, performance may be further improved by incorporating other criteria in the fitness function such as stability, smoothness, robustness, the number of non-zero weights etc.

The continuous-level discrete-time linear genetic network model employed in this paper was chosen especially for its simplicity and as a consequence has many shortcomings, such as the fixed time-lag and its linearity constraint. In addition, the results shown were based on data generated by a similar linear model, albeit corrupted by measurement noise. Nevertheless, our approach is one of first that has shown to work under the same conditions as can be found in a realistic application, i.e. for very large networks using data with very limited time-points and under noisy conditions. Therefore, we believe this approach has strong potential and we are currently investigating its use on real data.

8. ACKNOWLEDGMENTS

This work was funded by the IMDS program of the Delft University of Technology.

9. REFERENCES

[1] T. Akutsu, S. Kuhara, O. Maruyama, and S. Miyano. A system for identifying genetic networks from gene expression patterns produced by gene disruptions and

overexpressions. In *The Ninth Workshop on Genome Informatics*, pages 151–160, 1998.

- [2] A. Arkin, P. Shen, and J. Ross. A test case of correlation metric construction of a reaction pathway from measurements. *Science*, 277, 1997.
- [3] A. Arnone and B. Davidson. The hardwiring of development: Organization and function of genomic regulatory systems. *Development*, 124:1851–1864, 1997.
- [4] T. Chen, V. Filkov, and S. Skiena. Identifying gene regulatory networks from experimental data. In *Proceedings of the third annual international conference on Computational molecular biology (RECOMB99)*, pages 94–103. Association for Computing Machinery, 1999.
- [5] T. Chen, H. He, and G. Church. Modeling gene expression with differential equations. In *Pacific Symposium on Biocomputing '99*, volume 4, pages 29–40. World Scientific Publishing Co., 1999.
- [6] P. D’Haeseleer. Reconstructing gene networks from large scale gene expression data. *Dissertation*, December 2000.
- [7] P. D’Haeseleer, S. Liang, and R. Somogyi. Genetic network inference: From co-expression clustering to reverse engineering. *Bioinformatics*, 16(8):707–726, 2000.
- [8] P. D’Haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Mining the gene expression matrix: Inferring gene relationships from large scale gene expression data. *Information Processing in Cells and Tissues*, 1998.
- [9] P. D’Haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Linear modeling of mrna expression levels during cns development and injury. In *Pacific Symposium on Biocomputing '99*, volume 4, pages 41–52. World Scientific Publishing Co., 1999.
- [10] N. Friedman, M. Linial, I. Nachman, and D. Pe’er. Using bayesian networks to analyze expression data. *Journal of Computational Biology*, 7:601–620, 2000.
- [11] A. J. Hartemink, D. K. Gifford, T. S. Jaakkola, and R. A. Young. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. In *Pacific Symposium on Biocomputing 2001*, volume 6, Hawaii, January 2001. World Scientific Publishing Co.
- [12] F. Holstege, E. Jennings, J. Wyrick, T. Lee, C. Hengartner, M. Green, T. Golub, E. Lander, and R. Young. Dissecting the regulatory circuitry of a eukaryotic genome. *Cell*, 95(5):717–728, 1998.
- [13] H. Jeong, A. Barabasi, B. Tombor, and Z. N. Oltvai. The global organization of cellular networks. In R. Gauges, C. van Gend, and U. Kummer, editors, *Proceedings of the 2nd Workshop on Computation of Biochemical Pathways and Genetic Networks*, pages 3–12, Heidelberg, Germany, June 2001. Logos Verlag Berlin.

- [14] S. Liang, S. Fuhrman, and R. Somogyi. Reveal, a general reverse engineering algorithm for inference of genetic network architectures. In *Pacific Symposium on Biocomputing '98*, volume 3, pages 18–29. World Scientific Publishing Co., 1998.
- [15] E. van Someren, L. Wessels, and M. Reinders. Information extraction for modeling gene expressions. In *Proceedings of the 21st Symposium on Information Theory in the Benelux*, pages 215–222, Wassenaar, The Netherlands, May 2000. Werkgemeenschap voor Information Theory in the Benelux.
- [16] E. van Someren, L. Wessels, and M. Reinders. Linear modeling of genetic networks from experimental data. In R. Altman, T. Bailey, P. Bourne, M. Gribskov, T. Lengauer, I. Shindyalov, L. T. Eyck, and H. Weissig, editors, *Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology*, pages 355–366, La Jolla, California, August 2000. AAAI.
- [17] E. van Someren, L. Wessels, and M. Reinders. Genetic network models: A comparative study. In *Proceedings of SPIE, Micro-arrays: Optical Technologies and Informatics*, volume 4266, San Jose, California, January 2001.
- [18] E. van Someren, L. Wessels, M. Reinders, and E. Backer. Robust genetic network modeling by adding noisy data. In *Proceedings of the 2001 IEEE - EURASIP Workshop on Nonlinear Signal and Image Processing*, Baltimore, Maryland, June 2001.
- [19] M. Wahde and J. Hertz. Coarse-grained reverse engineering of genetic regulatory networks. In *IPCAT 99*, 1999.
- [20] D. Weaver, C. Workman, and G. Stormo. Modeling regulatory networks with weight matrices. In *Pacific Symposium on Biocomputing '99*, volume 4, pages 112–123, Hawaii, January 1999. World Scientific Publishing Co.
- [21] X. Wen, S. Fuhrman, G. Michaels, D. Carr, S. Smith, J. Barker, and R. Somogyi. Large-scale temporal gene expression mapping of central nervous system development. In *Proceedings of the National Academy of Sciences of the USA*, volume 95 of 1, pages 334–339. National Academy of Sciences, 1998.
- [22] L. Wessels, E. van Someren, and M. Reinders. A comparison of genetic network models. In *Pacific Symposium on Biocomputing 2001*, volume 6, pages 508–519, Hawaii, January 2001. World Scientific Publishing Co.
- [23] P. Woolf and Y. Wang. A fuzzy logic approach to analyzing gene expression data. *Physiological Genomics*, 3:9–15, 2000.