Abstract

Recent developments in microarray hybridization techniques have provided the opportunity to measure the activity level of thousands of genes using a single array. The behavior of gene activity during a specific process in an organism can now be monitored using a set of consecutive arrays. In order to understand the observed behavior of gene activity, one needs to determine the underlying regulations between genes that cause it. For this task gene interactions are modeled as a network of interactions which are learned from the gene expression data set.

The basic property of the time course data is that it provides measurements of a large number of genes taken over relatively few time points. This limitation makes it particularly difficult to accurately learn the network parameters. Fortunately, this limitation is balanced by the fact that genes are believed to interact with a limited number of other genes.

In this paper, a method for modeling relations between genes from a given data set is introduced that employs this limited connectivity. The method uses a genetic algorithm to find the structure and employs linear algebra to determine the weights of a linear genetic network. These inferred networks provide a complete description of the regulations between genes that accurately explains the measured time-course data.

1. Introduction

The ultimate goal of genetic network modeling is to find relationships between genes from a given data set. This goal is achieved by determining the parameters of a genetic network model such that the existence as well as the strength of a relation is determined. Figure 1 gives an example of such a genetic network. Lines represent the regulations between genes in two consecutive time points.

A large variety of different approaches, like Boolean networks [8], Bayesian networks [5,6], Quasi-Linear networks [3], Neural networks [15] and Differential equations [2] have been suggested to infer genetic networks from the data. These approaches can be divided into continuous and discrete models. In discrete models, both the relationships as well as the gene activity levels themselves are represented by discrete values. In the simplest discrete models, i.e. the Boolean networks, one allows only the presence or an absence of a relation.

Clearly, the results of such discrete methods could be sensitive to the discretization process. In contrast, continuous models represent relationships with a continuous value, allowing also the strength of the relation to be modeled. In general, however, the more complex the model is, the more parameters it has to learn, and the more difficult it will be to find the right solution.

Typical gene expression data sets consist of many genes, measured over a limited number of time points. Extracting relations consistent with all time points from such a time-limited data set results in a dimensionality problem, i.e. there exist many solutions that fit the data perfectly. To reduce the solution space as much as possible, we propose to use the simplest continuous models, i.e. a linear model [3] [12], to define the dependencies between genes in two consecutive time points. D'haeseleer showed in [3] a MSE (mean squared error) solution to this problem where he used interpolation to overcome the dimensionality problem. Van Someren [12] presented a clustering solution to reduce the number of signals in the given data set and a linear model was used to extract the dependencies between the clustered signals. In Wahde and Hertz [14], a genetic algorithm (hereafter GA) is used to estimate the relation matrix.
The model of Wahde and Hertz is most similar to the work presented in this paper. We also use a GA to solve the inverse problem. However, in contrast to Wahde and Hertz, we do not use the GA to solve the weights themselves directly, resulting in a huge search space. In particular, the GA is used to generate sparsely connected networks that then can be solved efficiently using Linear Algebra techniques. We will show that in this way the search space is restricted and as a consequence, convergence is achieved much faster and thus, in general, a more optimal result can be achieved.

In the next section, a complete description of the problem is given. The linear model is introduced that is used to describe the genetic network. Further, the pseudoinverse procedure for computing the weights of such a linear model is explained. A filter that, the constraints we propose to use on the linear genetic network are introduced, i.e. the simultaneous data fit and sparseness requirements. Further, we show that these constraints can be met by maximizing a fitness function. In section 4, we show how a GA can be used to perform this maximization effectively. Details are given about its behavior and characteristics. The results of our model as well as the conclusions are discussed at the end of this paper.

2. The Problem

The linear model for the genetic network used in this work is described by Eq. (1).

\[
x_i(t) = \sum_{j=1}^{N} w_{ij} \cdot x_j(t-1)
\]

In this basic equation, the activity level, \(x_i(t)\), of gene \(i\) at a certain point \(t\), has a weighted dependence with the activity level of all others genes at the previous time point. Weights \(w_{ij}\) represent the degree of influence of gene \(j\) on gene \(i\). The absolute value of these weights represent the strength of dependence between two connected genes. An absence of dependence is represented by a zero weight.

Suppose now that the given data set has \(T\) time measurements for the activity level of \(N\) genes. Eq. (1) results then in a system of equations where the \(w_{ij}\) are the unknowns to determine. For example, Eq. (2) represents the situation when only gene \(i\) is considered and \(T\) time points are measured.

\[
x_i(1) = w_{i1} x_1(1) + w_{i2} x_2(1) + \ldots + w_{iN} x_N(1)
\]

This set of equations can be written in matrix notation by Eq. (3).

\[
x'_i = w_i \cdot X^-
\]

Where \(x'_i = [x_i(1), \ldots, x_i(T)]^T\), \(w_i = [w_{i1}, \ldots, w_{iN}]^T\) and \(X^- = [x_1', \ldots, x_M']^T\), \(x_j' = [x_j(1), \ldots, x_j(T-1)]\) and the t indicates the transpose.

These \(T-1\) equations give information about the \(N\) weights in \(w_i\). Depending on the values for \(T\) and \(N\), different solution approaches need to be followed. In fact, three situations, namely: 1) \(T=1=N\), 2) \(T=1<N\) and 3) \(T-1>N^1\), each of them are discussed in more detail in next section.

2.1 \(T=1=N\): Exact solution

Assuming that the equations are linear independent, the exact solution can be simply found by Gaussian elimination [11].

2.2 \(T=1<N\): Over determined system

In this situation, a solution to the system that perfectly fits the data does not exist. Consequently, we have to look for the solution that gives the best approximation. Hence, techniques for minimizing the error that corresponds to a

\[1^1\] This situation applies only in the case of linear independent rows in the data set. In the case of dependency \(N\) should be replaced by the rank of the data set \(X\).
a certain solution have to be used. The error criterion that is generally used is the mean squared error (MSE) criterion as defined by Eq. (4). It measures the error as the distance between the predicted and the expected values for the activity level of a gene for all predicted time points.

$$\text{MSE}_i = \sum_{t=1}^{T} \left( w_i \cdot X^{-1}(t-1) - x_i(t) \right)^2$$  \hspace{1cm} (4)$$

Where $w_i$ are the unknown weights that influence gene $i$, column vector $X^{-1}(t-1)$ is the $(t-1)$th column of $X^{-1}$ and $x_i(t)$ is the activity level of gene $i$ at time point $t$. When solving such an over-determined system (i.e., finding $w_i$), the pseudoinverse procedure is used to determine the solution with the lowest MSE solution.

The basic steps in solving such a system are shown here. For sake of clarity we concentrate on Eq. (3):

$$x_i^* = w_i \cdot X^{-1}$$  \hspace{1cm} (5)$$

In an over-determined situation, matrix $X^{-1}$ has more rows than columns, making it impossible to obtain the solution for vector $w_i$ by multiplying both sides of Eq. (6) by $(X^{-1})^{-1}$ because this inverse does not exist.

Nevertheless, we can multiply both sides by $(X^{-1})^{	op}$, which is the transpose of $X^{-1}$:

$$x_i^* \cdot (X^{-1})^{	op} = w_i \cdot (X^{-1})^{	op}$$  \hspace{1cm} (6)$$

This ensures that $X^{-1} (X^{-1})^{	op}$ is always square and often nonsingular and hence, its inverse can be computed. By multiplying both sides of equation (6) by the inverse of $X^{-1} (X^{-1})^{	op}$ we obtain the solution for the weight vector:

$$w_i = x_i^* \cdot (X^{-1})^{	op} (X^{-1})^{-1}$$  \hspace{1cm} (7)$$

The pseudoinverse is in fact the definition of this solution that is:

$$\text{pinv} (X^{-1}) = (X^{-1})^{	op} (X^{-1})^{-1}$$  \hspace{1cm} (8)$$

Figure 2 gives a schematic illustration of the situation described above.

2.3 $T-1<N$: Under-determined system

In the under-determined case we, in fact, do not have enough information (measurements) to determine all the parameters (the $N$ weights) in the system. Hence, multiple solutions will exist that all fit the data perfectly. The description of these multiple solutions can be found as a combination of a particular solution $P_i$ and a homogeneous one $H_i$ [13]. In other words, the set of weights $w_i$ that perfectly fit the data can be written as:

$$w_i = P_i + \mu \cdot H_i$$  \hspace{1cm} (9)$$

Where the particular solution $P_i$ is one of the possible solutions to our system and $\mu H_i$ represents the ambiguity present in the data. $H_i$ is referred to as the null space.

Figure 3 visualizes the existence of multiple solutions for $T-1<N$. It represents the case where gene $i$ is only regulated by two other genes $k$ and $l$ and only two measurements are given. The information provided by the measurements thus restricts the two-dimensional continuous space of $w_k$ and $w_l$ to values on a line in the outcome space. The line is described by Eq. 9 where $H_i$ is the direction vector of the line and $P_i$ is an arbitrary point on this line.

2.4 $T-1<N$: Solved with pseudoinverse

It has been shown [9] that using the pseudoinverse from Sec. 2.2 in the under-determined case when dealing with a linear network gives also one solution on the line of possible solutions. This solution can be characterized by the fact that the squared sum of the weights is minimized.
In figure 3 this is the point on the line of possible weight values that is closest to the origin. If we would have been interested in the most sparse solution i.e. maximum number of the weights set to zero, then two other equally sparse solutions exist namely $S_1$ and $S_2$.

3. The M o d e l

In the previous section, a set of different situations that may occur was introduced. Our basic problem is to solve the system of equations as defined in Eq. (3) for all genes. In general, a given data set will have much more genes than time points [4]. In other words, we are in the situation of a highly under-determined system. Consequently, there will exist many solutions and we do not know which solution to choose.

Fortunately, we can employ constraints provided by biological knowledge. It is assumed that each gene is regulated by only 4 to 8 other genes [1]. This makes it possible to drastically reduce the number of weights to be estimated within our model. In fact, to such an extend that we may even arrive into an over-determined system. In this new situation, our genetic network thus contains only a limited number of connections and we can now use the pseudo-inverse procedure to determine the best approximation to the solution (like in Sec. 2.2).

Enforcing the biological constraint implies that we make the genetic network sparse. The problem that remains, however, is which connections (weights) need to be removed (set to zero) from the original fully connected network described in Eq. (1). This results in a combinatorial problem that we aim to solve through the maximization of a fitness function. This fitness function expresses the correctness of a solution with respect to the sparseness as well as with respect to the data fit (based on the M SE of the solution) of that sparse solution.

The fitness function should therefore combine information extracted from the data with biological information in order to provide the best solution with only a small amount of non-zero weights. The fitness function employed in this work is defined as:

$$\text{fitness} = f_D + \lambda \cdot f_S$$

The first term, $f_D$, is a measure of how well the solution can predict the data. The second one, $f_S$, has a biological meaning, and takes into account the number of zero weights (sparseness) in the solution. In the following, we go into more detail on the definition of both terms. Clearly, since we have defined a fitness function, other biological knowledge that one might want to exploit can easily be employed by introducing additional terms in the fitness function. Before explaining both terms, we would like to emphasize that parameter $\lambda$ plays an important role. This parameter is responsible for balancing between how good the solution is, in terms of the data fit, and the sparseness of the solution (i.e. the biological constraint).

The term that takes into account the sparseness of the solution is defined as follow:

$$f_S = \frac{\# \text{zeros}}{M}$$

Where, $\# \text{zeros}$ is the total number of zero weights in the solution and $M$ is the maximum number of allowed non-zero weights. We have defined such a limitation on the number of genes that can influence a gene in order to make sure that we always have more equations than unknown weights, or in other words, we always deal with an over-determined system. Hence, $M < T - 1$, where $T$ is the number of time points in the given data-set.

In order to let the first term represent how well the data is fitted we have adopted two different definitions for $f_D$. Table 1 shows both: inverse proportional dependence (IPD) and logarithmic dependence (LD). Clearly, both terms depend on the M SE of the sparse solution. The definitions differ in the way $f_D$ depends on this error.

Table 1: Two definitions for the first term, $f_D$, of the fitness function.

<table>
<thead>
<tr>
<th>$f_D$ Dependence</th>
<th>$f_D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{1 + \text{M SE}^{\beta}}$</td>
<td>Inverse proportional (IPD)</td>
</tr>
<tr>
<td>$\log \left( \frac{1}{\text{M SE}} \right)$</td>
<td>Logarithmic dependence (LD)</td>
</tr>
</tbody>
</table>

The difference between the two solutions is that the IPD, tends to one when the error tends to zero, while the LD tends to infinity. Consequently, the IPD scores solutions that are all equally well (all will have a score close to 1). The LD, on the other hand, scores these solutions quite differently since we operate then in the area where LD returns high values. Depending on the data set to analyze, we will choose one or the other function. In Sec. 5, both functions are tested in several experiments.
Now that the fitness function is defined, our purpose is to perform its maximization. The higher the fitness is, the lower the error of the solution and the lower the number of weights. Traditional gradient descend methods for finding the maximum of this function are not optimal since the function reveals several local maxima and a discrete shape. For this reason, we used a genetic algorithm to carry out the optimization of this fitness function.

4. Genetic Algorithms

A genetic algorithm (GA) optimizes the fitness function by parallel evaluation of some points in the search space. From these evaluated points, new points to be evaluated are derived by combining them and randomizing them. More specifically, a GA works as follows: 1) A number of different solutions are generated at random. These solutions are called chromosomes and form the set of solutions (a population); 2) For each of these chromosomes, the fitness function is calculated, that determines how good a solution is; 3) A new set of solutions is created from the initial population by combining (crossover) and randomizing (mutation) them. Solutions with better fitness have more chances to be selected to undergo crossover and mutation. Considered as consecutive steps, populations are also denoted as generations. 4) The steps are repeated along generations until we have reached a well enough solution, i.e. the fitness level of a chromosome becomes larger than a predefined threshold or a maximum number of generations has been reached. For a more complete description of genetic algorithms see [7] and [10].

The most important decision that has to be made when using GAs is how to encode the solutions of the problem into the chromosome. In a GA, a chromosome is a representation of one solution. Assuming that output genes in the genetic network have no dependence between them, we use separate GAs for finding the best combination of inputs for each of the output genes. In other words, the system of equations represented in Eq. (2) are solved for each of the N genes in our network separately.

In principle, we could use the representation described by Wahde and Hertz in [14]. With this representation, the elements in the chromosome are the weights that regulate the dependence between gene i and all other genes present in the genetic network, i.e.

\[ C_p^i = [w_{i1}, w_{i2}, \ldots, w_{in}] \] (13)

Where \( C_p^i \) describes chromosome p for gene i. The chromosome length thus equals the number of genes. The major problem of this representation is that the search space of solutions is very large (i.e. \( O(2^n) \) when each weight is encoded with 8 bits) and thus the inference will be very slow.

Instead of this codification, we can use a binary codification where the positions in the chromosome are boolean values which represent the existence or absence of a connection between two genes. From each connectivity vector described by a chromosome, weight values are calculated by Linear Algebra using the pseudo-inverse procedure. Eq. (14) shows an example of a chromosome using this representation.

\[ C_p^i = [b_{i1}, b_{i2}, \ldots, b_{in}] \] (14)

where \( b_{ij} \in \{0,1\} \) and represents whether gene j influences gene i (1) or not (0). The introduction of the separate Linear Algebra step improves the computation time of the algorithm drastically and more over the GA converges more easily since the new search space has become much smaller (in fact \( O(2^n) \), see also Fig. 4). However, this codification has still a major drawback (that the codification of Wahde and Hertz also suffers from). Namely, the constraint on the limitation of the number of allowed connections between genes is not always met with this codification since all weights are allowed to change (become non-zero). It may happen that in the case we start with an initial population that satisfies the limited number of non-zero weights, then, after a number of crossovers and mutations it is well possible that the number of non-zero weights can no longer be controlled. Consequently, the sparseness of our network is not fully exploited by this codification.

\[ \begin{bmatrix} W_{i1} & W_{i2} & \cdots & W_{in} \end{bmatrix} \]

Figure 4: Space of solutions for the different codifications in the GA.

\[ \begin{bmatrix} \text{Integer} \\ \text{Binary} \end{bmatrix} \]

\[ \begin{bmatrix} \text{Binary} \\ \text{Integer} \end{bmatrix} \]

\[ \begin{bmatrix} \text{Weight} \\ \text{Binary} \end{bmatrix} \]

\[ \begin{bmatrix} \text{Weight} \\ \text{Integer} \end{bmatrix} \]

\[ \begin{bmatrix} \text{Weight} \\ \text{Integer} \end{bmatrix} \]

An output gene is a gene at t+1 that is influenced by the activity of genes at the previous time point t, denoted by input genes.
We have implemented a new codification, which defines a chromosome length equal to the number of allowed connections. Instead of coding for each gene j whether it is connected to gene i or not, we now only encode the \( M \) connected genes to gene i, thus:

\[
C^i_p = \begin{bmatrix} j \ldots \ldots j \end{bmatrix}
\]  

(15)

Where \( j \) represent that gene \( j \) influences gene \( i \). This codification allows that gene \( i \) is maximally influenced by \( M \) genes (no more connections can be coded by \( C^i_p \)), hence the constraint about the maximum number of non-zero weights is always met. In the case that gene \( i \) is influenced by \( M1 \) genes (\( M1 < M \)), then \( M-M1 \) elements in the \( C^i_p \) will become zero.

Clearly, the advantage of this integer representation, with respect to the other representations, is that the constraint is always met. As a result, a much faster convergence of the GA is possible because the search space is dramatically reduced (in fact, this becomes \( O\left(\frac{M+N}{M} \right) \)).

Figure 4 shows the solution spaces.

A block diagram description of the major steps in the proposed method for modeling sparse genetic networks is presented in Figure 5. From a data set and with the information provided by the chromosome generated by the GA, a solution is computed using the pseudoinverse procedure. The solution's MSE and the number of non-zero weights (sparseness) are used by the fitness function to give a measure of the goodness of that chromosome. Along generations, solutions with better fitness are selected such that, after crossover and mutation, the new solution will eventually become better in terms of its fitness. The GA controls the space of solutions so that the best solution to the weight matrix is given at the end of the process. Note that, by selecting the maximum number of non-zero weights \( M \) smaller than the number of time points \( M < T-1 \), we in fact deal with an over-determined system.

5. Results

In this section, results obtained with the introduced method are presented. The first subsection introduces some basic definitions that are used to estimate the goodness of the results, i.e., inferential power IP, predicted power PP and computational cost CC.

In the next subsections, different experiments are carried out to study, firstly, the setting of the model parameters. Once the proper parameter settings have been established, the performance of this method for different characteristics of the data set are analyzed.

5.1 Basic Definitions

Inferential Power, (IP): This parameter measures the similarity between the predicted weight matrix \( \hat{W} \) and the true weight matrix \( W \).

\[
IP\left(\hat{W}, W\right) = 0.5 \left(1 + \phi\left(\hat{W}, W\right)\right)
\]

(16)

Here, \( \phi \) is the Pearson product moment correlation, causing the IP to be scaling independent.

Prediction Power, PP: This parameter reflects the similarity between the measured data set and a data set that is generated by using the predicted weight matrix given the same initial state. The definition of the PP is given by:

Figure 5: Schematic overview of the major steps in the proposed method as described in Sec. 3 and 4 for a given 5-gene data set with 4 time points.
Where,

\[
E_{HEP}(X, X^p) = \frac{1}{TN} \sum_{i=1}^{N} (x_i(t) - x_i^p(t))^2
\]  

(18)

In Eq. (18), \(X\) represents the measured data set, \(X^p\) is the generated data set from \(W^p\), and \(T\) and \(N\) are the number of time points and the number of genes respectively.

5.2 Setting model parameters

The proposed method requires four basic parameters to be set. The first one is the lambda parameter, \(\lambda\), presented in the fitness function. The other three belong to the GA, i.e. 1) chromosome length \(M\), 2) population size \((Pop)\) and 3) number of generations \((Gen)\).

To determine the proper settings of these parameters, several artificially generated data sets were used. In these experiments, the true weight matrix \((W)\) is repeatedly randomly generated. To ensure that these matrices are stable, the eigen-values are calculated and all elements in the weight matrix are divided by the maximum eigenvalue. The stable weight matrices are consequently used to generate many different stable data sets along \(T\) time points by using random initial states. By using such artificial networks, we know beforehand the solution for the weight matrix.

In the following experiments, a 15 genes weight matrix is used. The number of connections is set to 7 for all rows. The data set used to infer this weight matrix has 9 time
points (thus under-determined with respect to N and over-determined with respect to M). Each experiment was carried out with 8 different weight matrices and subsequent 8 data sets derived from them. The results are averaged over the 8 experiments. In the next figures, this average is represented by a line and the gray area represents the significant area, i.e. 2 times the standard deviation.

5.2.1 Setting lambda \( \lambda \) parameter

As mentioned before (Sec. 3), the \( \lambda \) parameter has a crucial role. With this experiment, we want to see how sensitive the fitness function is to this parameter. Several choices for this parameter were used and the results for both, the IPD and LD fitness functions are presented in Fig. 6 and Fig. 7. We selected a large enough setting for the population and generation parameter so that its influence was minimized. Setting the M parameter is, however, more critical since its influence is also quite large. It determines the chromosome length in the GA, which has a direct relation with the search space of solutions. The larger this parameter is, the more possible combinations exist for the genetic network. We have set this parameter to 8 from the know ledge that the maximum connectivity is 7 limiting as much as possible the search space. In Sec. 5.2.2 the influence of the M parameter is studied in more detail.

Once the other parameters have been set to an acceptable optimal value, experiments varying \( \lambda \) were carried out for both fitness functions. For the IPD function, shown in Fig. 6, the interesting area is in between 0.03 and 1, where the mean inferential power is close to 0.8 and the stability of the solutions is closest to 1 (see Fig. 6). From this we concluded that an optimal value for \( \lambda \) equals 0.05 when using the IPD fitness function.

For the LD function (see Fig. 7), the interesting area, where the inferential power is more optimal, is around \( \lambda \) equal to 30. For the remaining experiments, we have selected a value of 35 for \( \lambda \). Although the inferential power is worse than for the IPD fitness function, the stability of the solutions shows a more stable behavior using the LD fitness function (see Fig. 7).

In both cases, for \( \lambda \) equal to zero the inferential power is quite good (around 0.7). The reason for this is that we have set the maximum length of the chromosome to M close to the interesting area.

Another important point is the performance of this method with noisy data sets. In practice, all data sets will have noise because of the experimental process with which they are obtained. To test this, we have added Gaussian noise to each element of the data set, with zero mean and a 0.01 variance to mimic the expected measured noise.

Fig. 8 shows how the inferential power is dramatically reduced when noise is presented in our data set. This effect has been observed almost all currently known models [16]. The \( \lambda \) parameter now has a much lower influence on the result, which can be noticed from the flat curves. We think that the latter is due to the fact that M has not been optimized for this noisy situation. What is important to ask here is that the LD fitness function seems to show better behavior in noisy conditions.

5.2.2 Setting M parameter

The M parameter defines a direct relation between the number of genes in the genetic network and the maximum number of allowed connections (sparseness of the genetic network). The larger we set this parameter, the bigger the search space for the GA becomes because more connections are allowed. So it seems that it is quite important to set this parameter in an appropriate way. To test this, several values for the M parameter were evaluated, since the true connectivity was known to be 7, it could be expected that a good value for M would be 7.

The inferential power is reached using both fitness functions around M = 10 (see Fig. 9). First of all, this is allowed since even if the number of allowed connections is larger, still the actual number of non-zero weights can be smaller because the pseudo-inverse procedure returns zero values for those weights that are not necessary in spite the GA has defined a connection. Moreover, the GA can also define zero-connections inside the chromosome. From these results, we conclude that it seems that the search space needs to be somehow bigger than really is necessary to allow the GA for enough flexibility during crossing over the solutions.

Something else that can be noticed from Fig. 9 is the difference between inferential power for both fitness functions when M is below the connectivity of the genetic network (M < 7). In this case, it is not possible to obtain a good result because the chromosome size is not big enough to contain all the presented connections. In this situation, the IPD function gives a much better result than the LD one. The reason lies in the differences between the shape of the curves that define both fitness functions. While the IPD function has a range of values between 0 and 1, the LD function has a much larger range of values and shows a more non-linear relationship with the M SE.

In the previous section, we optimized the \( \lambda \) parameter in the optimal case (M < 7), where it is expected that the values for the M SE are small. Now, however, there are an insufficient number of connections and a higher value for the M SE is expected. The values of the IPD function, due
to its linearity, do not change very much so that the previously derived setting for optimal $\lambda$ parameter can now still be used. Nevertheless, for the LD function, values change drastically and fall in a completely different range of values. Hence, the setting for the optimal $\lambda$ is not valid in this new situation. Consequently, the results for $M < 7$ are worse than with the other fitness function.

In the presence of noise (see Fig. 10) the $M$ parameter has a different influence. Similar to the result with the $\lambda$ experiment, the curves for the inferential power are quite flat. There is a relative maximum value for $M = 8$ and from that point onward, the inferential power does not show a significant variation.

The behavior of the $\lambda$ and $M$ parameters in the presence of noise is not an expected one. It looks like if the second term of fitness function does not influence the final solution. A reason for this behavior may be that we are trying to infer a weight matrix from a noisy data set using only 9 time points. We are in the limit between an over-determined system and an under-determined system. The noise makes it impossible to perfectly predict the weight matrix because the amount of time points is insufficient. Several solutions in the flat area of the inferential power curve were manually checked for similarity. These checks showed that the solutions were completely different also in their sparseness. Hence, the conclusion is that only some relationship between genes can be obtained with such a limited data set, and that these relationships are shared to some extent by all solutions independently of the number of the allowed weights.

From Fig. 10, we can see how the stability has a different behavior for both fitness functions when varying $M$. While for the LD function the stability is reached till $M = 10$, the IPD function only gives stable solutions for $M > 10$.

Like was concluded when studying the $\lambda$ parameter, the LD fitness function gives more stable solutions in the interesting area.

5.2.3 Setting Population size and number of Generations

These last two parameters have a completely transparent influence on the performance of the proposed method. The

![Figure 8: Inferential Power varying lambda parameter in presence of noise for both fitness functions.](image)

![Figure 9: Inferential Power varying M parameter for both fitness functions.](image)

![Figure 10: Inferential Power and Stability for both fitness functions varying M parameter and in presence of noise.](image)
more generations we use and the bigger the population size is the better the results are for an experiment. There is, however, an important limitation in setting these parameters freely. The computational cost required for the GA to reach a good solution increase considerably when favorable values for these two parameters are chosen. Consequently, we have to balance between how accurate the solution is and how much time the algorithm takes for reaching it.

In Fig. 11 and Fig. 12 results for both parameters are presented. For Fig. 11, the number of generations was fixed to 25, while for Fig. 12 experiment, the population size was set to 150. It is obvious that the bigger the population is and the larger the number of generations the better the inferential power is. For the IPD fitness function a perfect inferential power is reached when the population size is around 600 or when the generations used are more than 75. The time required in these cases is close to $10^3$ seconds, which makes the experiment quite long but gives us the security that a perfect solution can be reached. An approximate estimation of the computational time is given in next equation:

$$O(\beta \times Pop \times Gen)$$

Which defines the linear relation between the number of generation and the population size with the computational cost in seconds. The $\beta$ parameter is a constant that takes into account the time in seconds for an experiment with only 1 generation and a population size of 1. In our case $\beta=0.0625$.

In balancing between the computational time and the goodness of the solution, we have selected a population size of 150 and a number of generations of 35. From Fig. 11 and Fig. 12 we may conclude that the Inferential Power is always better for the IPD fitness function.

Table 2 shows the basic settings for the parameters using both fitness functions. The population size and the number of generations have been set to the same value in both cases to maintain an equal computational cost.
Table 2: Basic Parameters for no-noise conditions.

<table>
<thead>
<tr>
<th></th>
<th>$\lambda$</th>
<th>M</th>
<th>Pop</th>
<th>Gen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverse Dep.</td>
<td>0.05</td>
<td>11</td>
<td>150</td>
<td>35</td>
</tr>
<tr>
<td>Logarithmic</td>
<td>35</td>
<td>10</td>
<td>150</td>
<td>35</td>
</tr>
</tbody>
</table>

In the presence of noise, the following conclusions can be drawn. In principle, the $M$ parameter has to be set in a region as close as possible to the maximum connectivity. In spite of the fact that no experiments are introduced here showing the dependency between number of generations and population size in noisy conditions, it is expected that, also in this situation, the GA needs more generations and bigger population size to give a good prediction.

5.3 Variation in the number of time points

The amount of information provided by the data set depends strongly on the number of measured time points. Using the basic parameter setting discussed in the previous section, the next experiment shows the dependence of the Inferential Power and Predictive Power with respect to the number of time points.

In Fig. 13, the results obtained with the IPD fitness function are compared with a reference model. This reference model generates a random solution. From this figure, we can extract the following conclusions. In principle, when the number of time points is smaller than the maximum connectivity, in this experiment 7, we have not enough information to exactly predict the data and exactly find the weights matrix, $W$. The more time points we have, the more extra information is added to the algorithm, so that the Inferential Power and the Predictive Power have a better value. When there are 2 time points more than the maximum connectivity, the Inferential Power and the Predictive Power are perfect for all solutions. Hence, this method performs as expected and with enough information ($T>M+1$) we can obtain a perfect prediction (see Fig. 13).

5.3 Variation in the number of time points

The amount of information provided by the data set depends strongly on the number of measured time points. Using the basic parameter setting discussed in the previous section, the next experiment shows the dependence of the Inferential Power and Predictive Power with respect to the number of time points.

In Fig. 13, the results obtained with the IPD fitness function are compared with a reference model. This reference model generates a random solution. From this figure, we can extract the following conclusions. In principle, when the number of time points is smaller than the maximum connectivity, in this experiment 7, we have not enough information to exactly predict the data and exactly find the weights matrix, $W$. The more time points we have, the more extra information is added to the algorithm, so that the Inferential Power and the Predictive Power have a better value. When there are 2 time points more than the maximum connectivity, the Inferential Power and the Predictive Power are perfect for all solutions. Hence, this method performs as expected and with enough information ($T>M+1$) we can obtain a perfect prediction (see Fig. 13).

Figure 13: IP and PP for the Inverse Proportional Dependence fitness function varying the number of time points in the data set. In this case, no noise was added to the measurements.

Figure 14: IP and PP for the Inverse Proportional Dependence fitness function varying the number of time points in the data set. Now noise was afterwards added to the measurements.
On the other hand, in the noisy case, showed in Fig. 14, the performances are drastically reduced. In the underdetermined area (T<7), the results are similar to the case of no noise (as showed in Fig. 13). So, in this area the noise does not seem to make a strong influence in the prediction. For T>7, however, adding more time points to the data set (more information) does not really give better performance. Comparing Fig. 13 and Fig. 14, the differences are remarkable. The differences in the Predictive Power are even higher. The maximum value for the Inferential Power in the presence of noise is about 0.7.

5.4 Real Data set

This section describes the analysis that was carried out on the gene expression profiles extracted from the 2467 genes in the budding yeast S. cerevisiae [4]. Due to the amount of genes presented in this data set, a first process of thresholding was carried out to ignore genes that are not active [12]. Signals (genes) with an expression level which never exceed ±1.1 are not considered. After the thresholding, a data set with 276 significant genes remains. To this new data set, a clustering process is applied so that the 276 signals are grouped in 15 clusters. For each of these clusters, a prototypical signal was made by averaging the signals in the group. The method described in this paper was then applied to these prototypical signals and the resulting weight matrix solution is shown in Fig. 15. Although it is not known whether this solution is the correct one, some conclusions can be extracted.

Firstly, this solution has been tested to predict the data set with a very low error (high PP). Hence, at least this weight matrix is able to predict the data. On the other hand, the maximum connectivity was set to 8 for this experiment. The solution returns this connectivity for all rows what might seem peculiar at first sight (the same number of connections for all genes). However, due to the strong reduction from the initial number of genes to only 15 signals one may expect a larger connectivity because now a prototypical signal (multiple genes) can influence all other prototypical signals. Nevertheless, the estimated relation matrix shows that the amount of influence measured by the weights is very different from some genes to others (i.e. the size of the connected weights differ considerably between different prototypical signals). For example, we can see in Fig. 15 that genes 2 to 6 present a very strong influence on the remaining genes. However, genes 12 to 15 have relative influence to the other genes.

6. Conclusions

In this paper, a method for modeling genetic networks, which combines a genetic algorithm and linear algebra, was proposed. Biological knowledge about the number of relationships between genes was included to obtain the simplest solution (the sparsest one). With the inclusion of this information, we transformed an initial underdetermined problem into an overdetermined one, which is solved using standard MSE techniques.

The advantage of the proposed method is that it is possible to make a definition of the connectivity level in the network so that, instead of needing more time points than genes in the data set, the number of time points only needs to exceed the maximum connectivity. This constraint can be well met in real situations where it is expected that genes have a limited number of connections (around 10) and the available data sets usually have more than 12 time points.

Extensive results were done for a 15 genes artificial network. The results were quite promising even showing a perfect inferential power, IP, and predictive power, PP, when the number of time points in the data set is larger than the number of connections within the underlying network. Hence, we can say that this approach gives a solution to the problem in the noise free case.

Although the method is promising one of the drawbacks is the amount of different parameters that need to be predefined. Although we presented already an extensive study of the influence of each of these parameters, we still feel that it is necessary to deepen this study, especially in the case of different connectivity levels and for different number of genes in the genetic network.
An actual data was analyzed using the proposed method. Although no ground truth was known for this data and no biological interpretation was given, the results are quite promising. At least the proposed model achieved the expected prediction and sparseness.

Two fitness functions were presented and used in the experiments. The IPD one has revealed a better performance in IP and PP terms. However, the LD one gives solutions with a better stability. As a future remark, it is useful to study whether the stability can be included into the fitness function since stability is imperative in the context of the biological system.

In the presence of noise, the results obtained by the proposed approach are not satisfactory yet. This is, however, known to be a common problem in the other approaches as well. Because of the performance of the proposed method in the noise free case, we still strongly believe that we should be able to interpret real data sets when the amount of noise introduced during the data acquisition step is limited. Possible additional criteria can be added to more effectively deal with the noise in the data.

7. References


