Available online at www.sciencedirect.com
ScienceDirect
BioSystems 88 (2007) 76–91
www.elsevier.com/locate/biosystems

Evolving fuzzy rules to model gene expression

Ricardo Linden[a,*], Amit Bhaya[b]

[a] *FSMA-RJ, R Monte Elíseo S/N°, CEP 27943-180, Macaé, RJ, Brazil*
[b] *Department of Electrical Engineering, PEE/COPPE/UFRJ, P.O. Box 68504, CEP 21945-970, Rio de Janeiro, RJ, Brazil*

Received 5 October 2004; received in revised form 29 March 2006; accepted 21 April 2006

Abstract

This paper develops an algorithm that extracts explanatory rules from microarray data, which we treat as time series, using genetic programming (GP) and fuzzy logic. Reverse polish notation is used (RPN) to describe the rules and to facilitate the GP approach. The algorithm also allows for the insertion of prior knowledge, making it possible to find sets of rules that include the relationships between genes already known. The algorithm proposed is applied to problems arising in the construction of gene regulatory networks, using two different sets of real data from biological experiments on the *Arabidopsis thaliana* cold response and the rat central nervous system, respectively. The results show that the proposed technique can fit data to a pre-defined precision even in situations where the data set has thousands of features but only a limited number of points in time are available, a situation in which traditional statistical alternatives encounter difficulties, due to the scarcity of time points.
© 2006 Elsevier Ireland Ltd. All rights reserved.

Keywords: Genetic programming; Fuzzy logic; Microarrays; Reverse engineering; Gene regulatory network

1. Introduction

Fuzzy logic is based on fuzzy set theory and especially on the concept of a fuzzy set. Informally, a fuzzy set is a set with imprecise boundaries, in which the transition from membership to non-membership is gradual rather than abrupt. A fuzzy set F in a universe of discourse U is characterized by a membership function μ_F, which associates each element $u \in U$ with a grade of membership $\mu_F(u) \in [0,1]$ in the fuzzy set F. Note that a classical set A in U is a special case of a fuzzy set with all membership values $\mu_A(u) \in \{0,1\}$ (Hiirsalmi et al., 2000).

A fuzzy implication is viewed as describing a fuzzy relation between the fuzzy sets forming the implication. A fuzzy rule, such as "if x is A then y is B" is implemented by a fuzzy implication (fuzzy relation) which has a membership function $\mu_{A\to B}(x, y) \in [0,1]$. Note that $\mu_{A\to B}(x, y)$ measures the degree of truth of the implication relation between x and y. A set of related fuzzy rules forms a fuzzy rule base that can be used to infer fuzzy results in the form of fuzzy sets.

Fuzzy logic offers an appealing method for describing phenomena by a set of rules and data sets. These data sets relate directly to concepts used on a daily basis, such as "fast", "strong" or "high", while the rules express knowledge approximately the same way a human expert would. An example of a fuzzy rule would be "if the car is fast, then the force applied to the brakes is strong".

Given these characteristics, fuzzy rules are easy to understand, verify and extend, since they are very simi-

* Corresponding author. Tel.: +55 21 2526 2344; fax: +55 21 2260 6211.
E-mail addresses: rlinden@pobox.com (R. Linden), amit@nacad.ufrj.br (A. Bhaya).

0303-2647/$ – see front matter © 2006 Elsevier Ireland Ltd. All rights reserved.
doi:10.1016/j.biosystems.2006.04.006

lar to the way a person might express knowledge. This also makes them attractive for use in domains where experts are available and can seed the systems with a number of effective rules from the outset (Bentley, 1999).

The goal of this paper is to propose an algorithm that finds a set of fuzzy rules that could represent the actual regulation of gene expression performed in the cell. This problem is analogous to fuzzy control, because there is an unknown control process determining how each gene will be expressed. In fact, a major challenge in current fuzzy control research is learning good controllers for large-scale, non-linear systems with many input and output variables where no training data are available from an expert (Carse et al., 1996).

The optimization abilities of evolutionary algorithms (EA) could be used to develop a good set of rules to be used by a fuzzy inference engine and to optimize the choice of membership functions. This has been done in other situations, starting as far back as (De Jong and Spears, 1991) and, more recently, in Bentley (1999), Dounias et al. (2002) or Yang et al. (2003) and in the review work in Freitas (2003) for example. EAs are used in this paper as a way to find a fuzzy rule base that might explain phenomena at hand.

Evolutionary algorithms are inspired by Nature. The idea is to mimic the natural evolution of the species in order to create a new kind of search technique that is robust and intelligently seeks solutions in a possibly infinite search space (Mitchell, 1996). Some of the techniques that are part of this branch of computer science are genetic algorithms (GA), genetic programming (GP) and evolutionary programming (EP).

All evolutionary algorithms use a population of competing solutions subjected to random variation and selection for a specific purpose (Fogel and Corne, 2003) which is to evolve the population to one that contains a higher proportion of superior ("fitter") individuals. The fitness of each individual in the population (its quality) is a measure of how well that individual achieves the desired goal. The variation and selection are usually based on two operators, the crossover operator which combines two different individuals into a new one and the mutation operator, which randomly changes parts of one individual in order to increase diversity.

An evolutionary algorithm could be described by the following pseudo-code

```
Create Initial Population
While termination criteria not met
    Select from current population parents which will generate
    offspring
    Apply genetic operators to the selected parents and generate
      offspring
    Select next population from current individuals and
      generated offspring
End While
Present best solution(s)
```

In this algorithm, the termination criteria are usually time based (a number of generations has elapsed), quality based (a certain performance has been achieved) or stagnation based (the best individuals has not improved for a certain number of generations), while parent selection is usually based on a roulette approach, where the parents with highest evaluation ("fittest") correspond to a bigger fraction of the roulette wheel.

Therefore, in order to define an EA one must define the coding scheme (how each individual will be represented in the computer), the operators (both mutation and crossover and any other specific one that will be used), the evaluation or fitness function (i.e., a measure of the quality of the current solutions to the problem at hand).

In genetic algorithms (GA), a form of EA, each solution is encoded by a binary string or another simple structure called a chromosome. Genetic programming (GP), which is another form of EA, can be used to evolve programs to perform certain tasks (Koza, 1992). In GP, the simple chromosome structure is replaced by a tree structure, in which a solution is either an algebraic equation or a program based on the input variables (Yang et al., 2003).

Many problems of current interest in bioinformatics have high dimensionality without a corresponding number of examples (data points) that would allow the application of well-known statistical techniques. One such area is the reverse engineering of genetic networks, further described below. The data set available for this reverse engineering generally consists of hundreds to thousands of signals, usually measured for no more than twenty time-steps. The paucity of data renders network models inferred from this data statistically insignificant (Van Someren et al., 2000). Since GP methods are able to generate a broad spectrum of solutions, even from incomplete or insufficient data sets, they are adequate for application in this area, generating testable hypotheses for the biological laboratory. GP methods might even succeed when the data scarcity might make statistical methods unable to generate solutions.

The ongoing revolution in the field of genomics can be attributed mainly to the development of new tools that have flooded scientists with huge amounts of data. These new tools have made it clear that the current understanding of biological phenomena will not be impaired by lack of data, but rather by the growing difficulty in analyzing and interpreting it adequately. One of the technologies that is causing this huge impact in the biological sciences is DNA microarray technology, which consists of an ordered array of nucleic acids, proteins, small molecules, that enables parallel analysis of complex biochemical samples (Schena et al., 1995).

Microarrays are based on the idea that, in every cell, at every state, only a fraction of the total DNA is transcribed into mRNA, which is subsequently translated into protein. The translated gene is said to be expressed and the analysis of the expressed genes is called gene expression analysis.

Microarrays allow one to study expression levels in parallel thus providing static information about gene expression (i.e., in which tissue(s) the gene is expressed) and dynamic information (i.e., how the expression pattern of one gene relates to those of others). The high degree of digital data extraction and processing of these techniques supports a variety of samples or experimental conditions (Duggan et al., 1999). Microarray technology, as a high throughput approach of differential gene expression studies, efficiently generates massive amount of gene regulation data, facilitating scientists in quickly identifying what gene candidates to follow up with functional characterization.

In higher metazoa, each gene or protein is estimated on average to interact with four to eight other genes (Arnone and Davidson, 1997), and to be involved in about ten biological functions. Therefore, the expression process in any particular cell can be seen as a dynamic network, the nodes of which are genes and the links between genes having real-valued weights, either positive or negative, which model the degree to which the expression of one gene affects the expression of another (Fogel and Corne, 2003).

This is a simplified view of gene expression, appropriate for exploratory data analysis. In reality, gene expression is a complex process regulated at several stages in the synthesis of proteins. Apart from the regulation of DNA transcription, the best-studied form of regulation, the expression of a gene may be controlled during RNA processing and transport (in eukaryotes), RNA translation, and the post-translational modification of proteins. The degradation of proteins and intermediate RNA products can also be regulated in the cell. The proteins that carry out the above regulatory functions are produced by other genes. This gives rise to genetic regulatory systems structured by networks of regulatory interactions between DNA, RNA, proteins, and small molecules (Smolen et al., 2000).

The global gene expression pattern is therefore the result of the collective behavior of individual regulatory pathways. In such highly interconnected cellular signaling networks, gene function depends on its cellular context; thus understanding the network as a whole is essential. However, dynamic systems with very large numbers of variables also present computational difficulties.

It is not practical, especially under tight financial constraints, to propose that biologists should execute tens or hundreds of microarrays in order to develop a biological hypothesis. This is the context in which the algorithm proposed here should be used, i.e., to search for candidate regulators that could reduce the number of experiments needed for genetic network determination.

Other approaches to the problem of genetic network determination are now described briefly. For example, (Creighton and Hanash, 2003) searched for association rules, while (Ideker et al., 2000) and (Wagner, 2001) performed perturbation analysis on the genetic data and (Weaver et al., 1999) trained neural networks. All these methods are data intensive, and the process of obtaining these data sets is expensive. In this paper we propose a method for generating hypotheses that can guide experiments and perhaps decrease the number of experiments performed, allowing for an economy of time and money, and increase the efficiency of biologists as well as of the methods mentioned above.

The rest of this paper is organized as follows: in Section 2, we describe the algorithm proposed and in the Section 3.1 it is shown one possible application for it. In Section 3.2 it is discussed how this algorithm compares to others previously discussed in literature and finally, Section 4 contains concluding remarks and a discussion on some improvements that may be made in the future on the work proposed in this paper.

2. The proposed genetic program

Given the pseudo-code in Section 1 above, the GP proposed here can be further specified by its three basic elements: chromosome structure, evaluation function and genetic operators. The other elements used in the GP are described in the next section of this paper.

The model used in the prediction of control structures for microarray data analysis is described in this section.

2.1. Pre-processing

The genetic program described in this paper deals with the undetermined nature of microarray data, which leads to a “blessing of dimensionality”. An average microarray has hundreds, or even thousands, of genes measured in a single-digit number of time points. This leads to a situation where there are so many degrees of freedom and so few constraints, that there are many possible "good fit" solutions. Thus, it can be argued that any search method will stumble upon a reasonable solution, and the key is how to differentiate between various solutions.

In order to deal with this problem, a clustering approach was developed. This approach is based on the assumption that genes that exhibit the same behavior are under the same kind of control, which means that strongly correlated genes should be under the same control. Since, the goal of the work is to discover control relationships, this assumption leads to the consequence that any strategy that is found to work for one gene should work for all the ones that are in its cluster. Therefore, searching for the techniques that work in the entire cluster helps to eliminate many that would work solely on one of the genes in it.

The clusters were developed using the Pearson product-moment correlation coefficient (Moore and McCabe, 2005) as distance metric. A cut-off value was defined ad-hoc and all genes that have a correlation value with the gene under consideration above the chosen value were considered to be in the same group. One of the clusters generated is shown in Fig. 1.

The assumption that every gene is under the same control may not often be correct, but since the correlation value used as a cutoff is very close to 1, all genes have the same pattern of expression as the gene under consideration. Therefore, this approach will not create spurious associations, and will limit the number of regulations that will effectively fit the data correctly, therefore reducing the "blessing of dimensionality" that is inherent to microarray data.

The second operation performed to reduce the data dimensionality was to try to find the suitable candidates

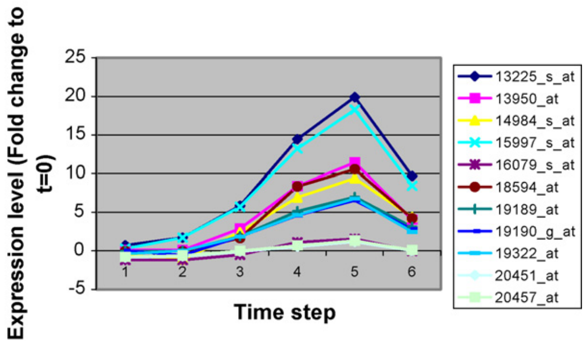

Fig. 1. Example of cluster generated when looking for the regulation of gene 15997_s_at. All the genes show a similar expression pattern over time, even if their expression values are not equal.

for the regulator role. In order to choose these candidates, the correlation coefficient was used once again. In order to infer causation from a high correlation, it is necessary to establish that change in one of the variables always occurs first and that the causation hypothesis is in accordance with a specific theoretical model. In other words, high correlation by itself does not imply causation.

It was decided then to use the delayed correlation, which is defined as the correlation between the expression values of the genes under evaluation at time t and the expression values of the candidate regulators at time $t-\tau$. An ad-hoc value of 0.80 was established as the cut-off value for the candidates that accounted for a reduction in data dimension of one order of magnitude. Only those genes that passed the cut-off criterium were selected as possible regulators and submitted to the genetic programming algorithm.

2.2. *Chromosome structure*

A rule is represented as an expression in reverse polish notation (RPN). In this notation, all operators precede their operands. An operand can be an expression that has its own operator, and so on, recursively. Three operators are used: NOT, OR, AND. Although every logical expression can be represented with just two (either NOT and AND or NOT and OR), using three makes the resulting expressions smaller, simpler and easier to understand.

RPN is well suited to a tree representation, in which operands are the descendants at the subtree rooted at the operator. Each operator has two descendants, with the exception of NOT, which has only one. This tree representation is, in turn, well suited to a graphical display, as shown in Fig. 2.

Yang et al. (2003); (Dasgupta and Gomes, 2001; Dasgupta and Gomez, 2002) are good examples of previous use of RPN notation and their use is compared to the one proposed here in Section 4 of this paper.

One important difference with respect to (Yang et al., 2003) is the fact that he does not allow for the use of the operator NOT, which would be extremely useful in his linguistic definition of cleavage rules, while (Dasgupta and Gomez, 2002) uses a representation that relies on bit strings, which could make the chromosomes very large, a problem that we experienced in our previous work (Linden and Bhaya, 2002a, b).

Another feature in which the GP proposed here differs from these previous works is the fact that the crossover operator used here, which is strongly based on the

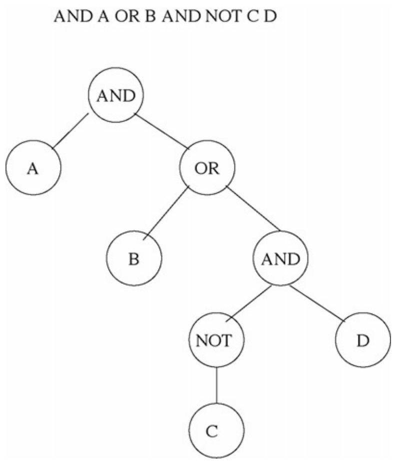

Fig. 2. Graphical representation of a typical rule consequent.

standard genetic programming operators, as defined in (Koza, 1992), is similar to a uniform crossover, whilst both works mentioned above use a slight variation of one point crossover.

(Bentley, 1999), like (Dasgupta and Gomez, 2002), also uses binary string representation as well as fuzzy sets without intersections. In addition to large chromosome size, as mentioned above, our experiments, on our data, with nonintersecting fuzzy sets indicated severely degraded performance, due to the fact that a single rule is active at any point.

(Zhou et al., 2002) also evolves rules with RPN. Nevertheless, he uses some mathematical operators such as square root, addition and multiplication that, although useful in modeling function, are difficult to justify in a cellular environment, where there is no evidence of such complicated mathematical operations. If other applications were the goal of the algorithm, these operators would be useful, but in the current context of genetic control, it is preferable not to use them on account of their lack of resemblance to real cellular behavior.

Whenever discovered knowledge is to be used to assist in decision-making by a human user, it is important that it be comprehensible to the user (Freitas, 2003). Since the main purpose is to create hypotheses that can be verified afterwards in the biological workbench ("wet lab"), simplicity in the rules is an essential feature.

In order to force the rules to be as simple as possible, a parameter is embedded in the rule initialization mechanism. This parameter is called the expression height coefficient, and the larger this coefficient is, the shorter the rules are.

This coefficient is used in the random initialization of the trees. When generating a new rule a number is randomly selected in the interval [0,1], and a hard limit is calculated according to the following formula:

$$\frac{1}{2\times(\text{coefficient}-1)} \qquad (1)$$

If the random number is smaller than this limit, an operator that allows the tree height to increase is selected. Otherwise, a fuzzy set is selected and a leaf is inserted in the tree.

Several different coefficient values were experimented with and it was found that the best results were achieved when its value was set to 3, meaning that the expression trees generated have an average height of 2.5. Tree height is also a factor in the evaluation function in order to give preference to simpler and shorter rules (see below, for further discussion of this topic).

The idea of representing regulatory networks as a set of rules is not new. It has already been explored in several papers, such as (Mcadams and Shapiro, 1995), where cell dynamics was compared with circuit dynamics and simulated using rules representable by logical gates and also in (Venet et al., 2001), where binary switches were defined and binary rules based on them were also used to discover useful relationships in gene regulation.

A problem that might arise in the use of rules is described in the experiment, reported in (Thomas, 1998), that adds a high amount of inducer to an uninduced culture of *E. coli*, immediately splits the culture into two parts (A and B) and dilutes them so that the extracellular inducer concentration drops to reach the "maintenance" range. The only difference between subcultures A and B is that in A dilution takes place immediately, whereas in B, it takes place after a delay of 10 min. Despite this, subculture A, which was not in contact with a high concentration of inducer for a significant time, is and remains uninduced. In contrast, subculture B is and remains induced, because it was in contact with a high concentration of inducer for enough time to be fully induced, and dilution to the maintenance concentration does not change the situation.

In order to keep them growing, subcultures A and B can be serially diluted, always in the same medium (with the maintenance concentration of inducer). In the experiment reported in (Thomas, 1998), this was done for 150 generations, after which A remained uninduced, B fully induced. The results were made even more striking by the demonstration that when a mixture of uninduced and induced cells is present in the maintenance medium, the progeny of the induced (respectively uninduced) cells is induced (respectively uninduced).

This could imply that more than one rule for each element in the gene network may be necessary – one for

Table 1
Interpretation of each fuzzy set associated to gene expression

	Number of fuzzy sets			
	2	3	5	7
Fuzzy sets names	Expressed Not expressed	Low expression level Medium expression level High expression level	Very low expression level Low expression level Medium expression level High expression level Very high expression level	Extremely low expression level Very low expression level Low expression level Medium expression level High expression level Very high expression level Extremely high expression level

each state it might assume. It is an important issue to create a mechanism that would ensure that the rules created would really deal with this characteristic of dependency on different time delays for different regulators, and therefore the algorithm proposed allows for this possibility.

It is assumed that each feature may be associated with several fuzzy sets, described in Table 1. This is different from the situation one would face in a discrete (Boolean) algorithm where the gene could be either expressed (binary value 1) or not (binary value 0). Obviously, since fuzzy logic is being used, membership can be different from 0 to 1 in any set at any given time.

The expression space of each feature is divided evenly, so that each fuzzy set has an equal support. The process is shown in Fig. 3. There is overlapping among adjacent fuzzy sets, so that more than one rule may be active for each expression value.

When we are using fuzzy logic, there is no need to normalize the expression data because each expression value will be turned into a membership value, which will be used in the rules. Linear transformations would be specially pointless, for the fuzzy sets would be exactly the same, but dislocated in space. A non-linear

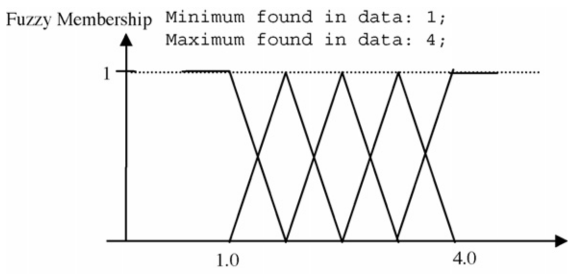

Fig. 3. Division of an expression space in five fuzzy sets. The interval is read from the data set and is divided in four parts. The first half part is the support of the first descending function, the last half part is the support of the last ascending function, while the three remaining parts are the supports of the three triangular membership functions.

transformation of the data, such as the application of a log transform is possible, especially in the case where there may be a large variation in the expression values of a gene. This was not a common occurrence in the data analyzed in this paper, since most variations of the expression values were of the same order of magnitude.

Each fuzzy set must be associated with at least one rule. Each rule antecedent is a tree similar to the tree depicted in Fig. 1 above. The only difference is that the elements that form the rule are now fuzzy sets for one variable, instead of the variable expression used before. The consequent of each rule is one fuzzy set of the regulated gene.

The average number of rules per fuzzy set is a parameter to be chosen. Nevertheless, it is not guaranteed that the number of rules per fuzzy set is equal to the average, since mutation, as used in the GP, may change the total number of rules per gene and per fuzzy set.

Hence, if a gene is associated with k fuzzy sets, and the average number of rules per fuzzy set is m, a gene will have a number close to $m \times k$ rules, where each of the rules is described by the following set of syntatic rules:

```
<rule>:: = IF <antecedent> THEN <consequent>
<consequent>:: = <Fuzzy_Set> (<gene_name>)
<antecedent>:: = <Fuzzy_Set> (<gene_name>)|NOT
  <antecedent>|AND <antecedent> <antecedent>|OR
  <antecedent> <antecedent>
```

Each fuzzy set is guaranteed to be associated with at least one rule, which ensures that it will cooperate in the calculation of gene progress. In the examples shown below in this paper, the average of rules per fuzzy set was defined as 3, while using three sets per element. Therefore, each of the chromosomes created represents a fuzzy rule base containing, on average, nine rules. Since gene names were used directly in the chromosome representation, the number of genes does not affect the chromosome size (as it would in a binary representation). Hence the chromosomes created are not very big and it is computationally efficient to evolve them.

Another important characteristic of the proposed algorithm is that it allows the user to insert rules that he may find interesting for the algorithm to research. These rules may be inserted as required, desirable or forbidden. In the first case, it is guaranteed that they will appear in the rule base and in the last case it is guaranteed that they will be banned from it. In the case of rule insertion as "desirable", a high percentage (25%) of the candidate chromosomes are initialized with the desirable rules and also give these candidates a high percentage (10%) chance of being chosen during mutation. This gives them a high chance of being present in the final rule base evolved.

This characteristic is an important step towards achieving the primary goal of this work, which is to offer biologists a tool that allows them to test and create some hypotheses in silico before resorting to expensive in vitro experiments for confirmation.

2.3. Evaluation function

In order to evaluate the performance of the chromosome, t trajectories with ns_t steps each were stored. Each trajectory represents the "real" behavior of the network to be modeled. Therefore, every network receives the first state of each trajectory and the GA calculates the intermediate and final steps for this network. When dealing with Boolean data, the number of bits that differ at each stage of the trajectory is added and averaged over the number of steps.

The averaging is necessary in order not to penalize long trajectories. It is clear that a small difference over a large number of steps will lead to a large sum. On the other hand, a large difference over a small number of steps will lead to a small sum. Thus errors are averaged in order to prevent the multiplicative effect of the number of steps.

This approach does not account for one fact that is biologically plausible, namely, that a gene may regulate another only at a certain time point, while remaining quiescent during the rest of the interval evaluated. In this scenario, a single high error rate at one time point is indicative of the fact that perhaps only at that time the regulation was wrong.

In order to be able to deal with such a situation, it would be necessary to define a different regulation function for each time point. This is not done in this paper: it is supposed that the same regulation function is always acting on a regulated gene, but such an improvement will be considered in future work.

In previous Boolean work (Linden and Bhaya, 2002a, b), each chromosome was penalized according to the number of active relationships it represents, so that the shorter chromosomes get extra credits. (Rutter and Zufall, 2004) gave empirical evidence that the length of a biochemical pathway has the potential to constrain the rate of evolution in that system. In the set of 48 sequenced organisms studied in their paper, there was a strong tendency for longer amino acid biosynthesis pathways to remain evolutionarily static in their structure while shorter pathways demonstrated greater evolutionary ability, suggesting that the connection of steps in long pathways may deter evolutionary change. Another interesting piece of evidence on this topic comes from (Wain-Hobson et al., 2003), where it is shown that in retrovirus, shorter pathways are dominant due to the massive recombination that takes place.

It is important to understand that there also seems to be some evidence that nature rewards redundancy in the form of alternative pathways performing the same function, so that there are backups in case that one pathway fails. This is often a problem in knock-out experiments, where knocking one seemingly crucial component of a pathway may not lead to the pathway being switched off, since a backup component is activated and takes over the function of the knocked one. Nevertheless, only the currently active pathway can be uncovered and the idea that shorter individual pathways are less prone to interruption seems to be true.

Since a bit string chromosome is not used in our approach, a different approach to reward minimality is adopted. A coefficient $c \leq 1$ is created to multiply by the evaluation of the chromosome. The bigger the chromosome tree height, the smaller is the coefficient c. Therefore, in the case where there are two different chromosomes with the same evaluation, the simpler and shorter one will be preferred. This coefficient is given by the following formula:

$$\begin{cases} c = \dfrac{1}{n}, & h \leq 2, \\ c = \dfrac{1}{(h-1) \times n}, & h \geq 2 \end{cases} \tag{2}$$

where n is the number of rules describing a regulation and h is the maximum height of the trees that describe the corresponding rules.

It is also important to recall that the number of time-points that are required to identify correctly the weights of a sparsely connected network is of the order of its maximal connectivity (Van Someren et al., 2000). Therefore, in the context of few available data points, it makes sense to look for networks with lower connectivity.

There are, of course, many ways to calculate the difference between the calculated trajectories and the ones read in the microarray data, which are composed of real values. In the discrete (Boolean) case described in (Linden and Bhaya, 2002a, b), the difference is absolute (either the bit is equal to the one read, or it is not), thus the measure adopted is to count the number of bits that differ from the supposedly correct ones in each step on each trajectory.

On the other hand, in the continuous case, there is a finite continuous difference whose meaning is different from element to element. For instance, if the read value is 0.1, a difference of 0.02 is quite high, while the same difference, when compared to a read value of 25.5 is almost negligible.

Therefore, instead of using the absolute difference, the mean absolute percentage error (MAPE) was used. This metric is defined by the following formula:

$$\mathrm{MAPE} = \frac{1}{N}\sum_{t=1}^{N}\left|\frac{y_t - \hat{y}_t}{y_t}\right| \tag{3}$$

This value is divided by the number of steps per trajectory and summed over the trajectories available and its inverse is used as the evaluation for the chromosome. The overall formula is:

$$\frac{1}{\left(\sum_{t=1}^{\text{no of traj.}}\left(\frac{\sum_{g=1}^{\text{no of genes}}\mathrm{MAPE}_g}{\text{no of steps}_t}\right)\right)/\text{no of traj.}} \tag{4}$$

The calculated value is inverted because this is an error minimization process and therefore, larger errors should receive smaller evaluations and, consequently, smaller portions of the roulette wheel.

This formula results in a number without units that does not have any clear relationship with any error measurement. In order to make its meaning clearer to the program user, a number called “infinite” was calculated, based on an error level set by the user for each gene in each step of each trajectory.

The evaluation is also inversely and linearly proportional to the per trajectory average error. If there is a single trajectory, there is an inverse relationship between the average trajectory fitting error in the trajectory and the chromosome evaluation function. Hence, a five times bigger error will have a five times smaller evaluation function.

For example, if there is one trajectory of 10 steps, the 0.5% error level generates an error measurement of 200, so that the user can understand the numbers generated by the evaluation function simply by comparing it to the “infinite” level previously calculated.

This evaluation function is appropriate because by measuring the absolute differences at each time point, it also captures the changes in time of the expression values. For instance, if an expression value increases from time t to time $t+1$ and the calculated value decreases in the same period of time, the error value at this time point will be very high. Therefore, every chromosome that has a high evaluation will calculate a solution that has high correlation with the function that maps real expression values changes over time.

2.4. Genetic operators

In order to complete the definition of the GP proposed, the mutation and crossover operators are now defined. As is typical in evolutionary algorithms that use competing operators, the GP proposed in this paper alternates between the mutation and crossover operators with a probability that is either fixed or time varying, depending on a user supplied parameter.

2.4.1. Crossover operator

The crossover operator is intended to exchange information between two different individuals in a way similar to sexual reproduction, and is similar in performance to uniform crossover for genetic algorithms.

The operator works by randomly choosing sub-trees to interchange between the chosen parents. Each node n_c of each tree is visited and a random draw decides whether or not the sub-tree rooted in n_c will be exchanged with the other parent. An example of this operator can be seen in Fig. 4. This strategy preserves the subexpressions rooted at the selected nodes and therefore the

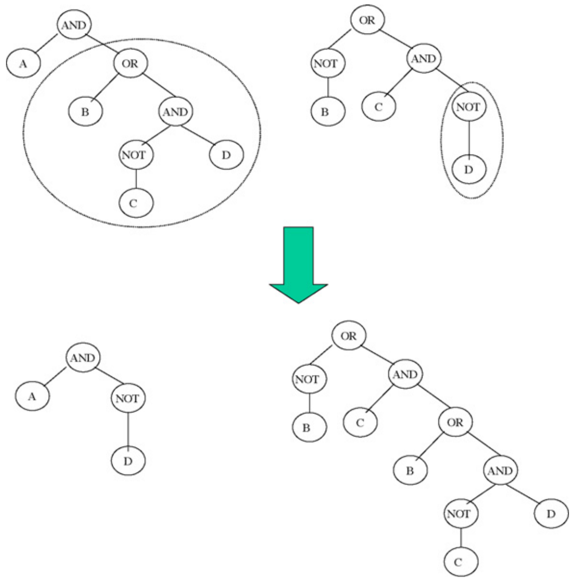

Fig. 4. Example of crossover action on our trees.

crossover is not too disruptive with regard to the current population.

Since there may be multiple rules per fuzzy set, there must be an additional control mechanism to choose which rules will exchange sub-trees. In this case, the crossover operator guarantees that the rules for a fuzzy set *X* in chromosome 1 (Chrom_1) only crosses with rules for the same fuzzy set in chromosome 2 (Chrom_2), even if there are more rules for this fuzzy set in one chromosome than in another. This means that if chromosome Chrom_1 has two rules for fuzzy set *X* and chromosome Chrom_2 has only one, the two rules from Chrom_1 will cross with the single one from Chrom_2. If the situation is reversed and Chrom_1 has only one rule while has Chrom_2 two or more, the number of rules in the resulting chromosome will still be equal to the number of rules in Chrom_1.

If both of them have more than one rule for fuzzy set *X*, a random choice will be made between the alternative rules, only ensuring that each rule crosses at least once. An example of this situation is shown in Fig. 5. Obviously, crossover generates two rules per operation, which are done applying the rules described here twice: first considering the order C_1/C_2 and secondly, the order C_1/C_2. Therefore, children with the characteristic structure of both parents will be generated.

Given that the evaluation function used evaluates each part of the chromosome separately (the regulation strategy for each node is given a separate evaluation), different quality measures for each node should be used. This means that a chromosome may be very good for regulating one node and very bad for the others.

Therefore, when selecting mates, a different mate may be chosen for each gene regulated. In fact, the crossover operator may also involve multiple chromosomes, as opposed to the classical approach of a pair of chromosomes. (Súer et al., 2002) used a strategy to force the worst chromosomes to mate before vanishing from the population. This could be considered in future implementations, although experiments have shown that genetic variability has been well preserved in the populations generated by the proposed algorithm. Future implementations of a parallel evolutionary algorithm might also be an interesting alternative to create good genetic variability.

2.4.2. Mutation operator

The mutation operator is actually threefold—there is rule mutation, insertion mutation and deletion mutation.

Rule mutation works in the way proposed while discussing the discrete mutation operator (Linden and Bhaya, 2002a, b). The only part that cannot be changed is the consequent; otherwise the rule would refer to a different fuzzy set.

The rule mutation randomly chooses one node in the rule tree and prunes the whole branch. Following this, a new sub-tree is generated using the same generator that created the initial population and the branch is then replaced. The population generator in this specific case is instructed to generate short trees (trees whose height is equal to or less than 3) by setting the expression height coefficient, as described in Section 2.1. Its modus operandi is described in Fig. 6.

Insertion mutation randomly chooses a fuzzy set for which to create a rule. The new rule is generated using the same random rule generator that generates the initial population, while deletion mutation chooses one rule to delete. It will be deleted if it is not the only rule for a fuzzy set.

The last two mutation operators described above change the number of rules per fuzzy set, making it possible for a certain gene to diverge from the average number of rules defined as parameter. Since their choice has the

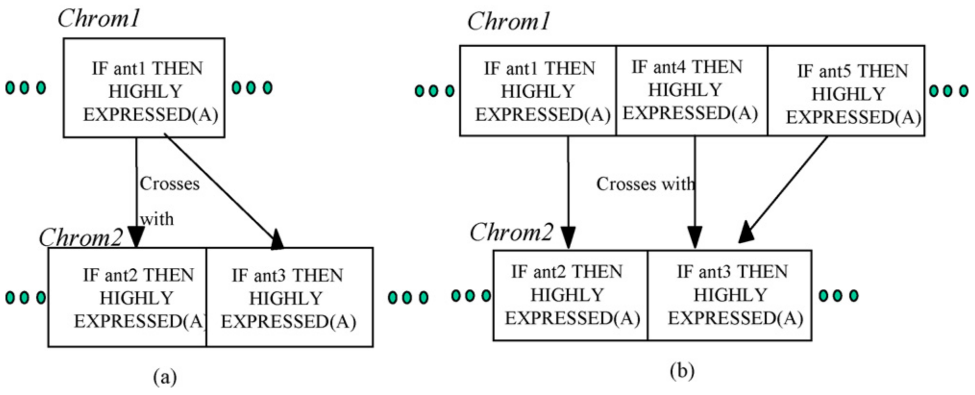
Chrom1
IF ant1 THEN HIGHLY EXPRESSED(A)
Crosses with
Chrom2
IF ant2 THEN HIGHLY EXPRESSED(A)
IF ant3 THEN HIGHLY EXPRESSED(A)
(a)
Chrom1
IF ant1 THEN HIGHLY EXPRESSED(A)
IF ant4 THEN HIGHLY EXPRESSED(A)
IF ant5 THEN HIGHLY EXPRESSED(A)
Crosses with
Chrom2
IF ant2 THEN HIGHLY EXPRESSED(A)
IF ant3 THEN HIGHLY EXPRESSED(A)
(b)

Fig. 5. Example execution of the crossover operator. In (a) we see the situation where chromosome 1 only has one rule for the set being analyzed, so this rule crosses with all the rules for chromosome 2. In (b) we see a situation where both chromosomes have more than one rule. The ones to cross are chosen randomly but the algorithm ensures that each rule will cross at least once.

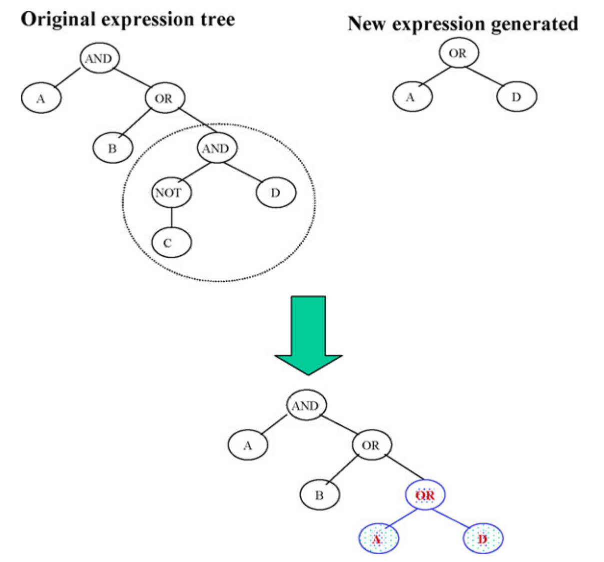

Fig. 6. The mutation operator chooses randomly a sub-tree to cut (circled) and replaces it with a newly generated short tree (dot filled circles, lower right corner).

same probability, the global average will not be affected – some genes will have their number of rules increased, others will have it decreased, and the global number of rules will almost always remain stable.

3. Application to genetic network determination

This section describes two applications of the proposed algorithm to real, continuous data originating from microarray experiments. In previous work, Boolean data has already been analyzed by this algorithm (Linden and Bhaya, 2003), showing that it is possible to mix both kinds of data using the algorithm proposed here.

3.1. Methodology

In the GP proposed three different termination criteria were used:

- one based on number of generations (no more than 80 generations per run),
- one based on stagnation (stop the run if the best solution stagnated for the last 20 generations),
- one based on the quality of the solution found (stop if the data was fit to a maximum error of 1%.

The limited number of generations was determined empirically. It was somewhat above the limit where genetic convergence could be perceived in the population. The effect of genetic convergence can be seen in Fig. 7, where we show the evolution of the fitness for one of the examples described below, the determination of the rule set for preGAD67.

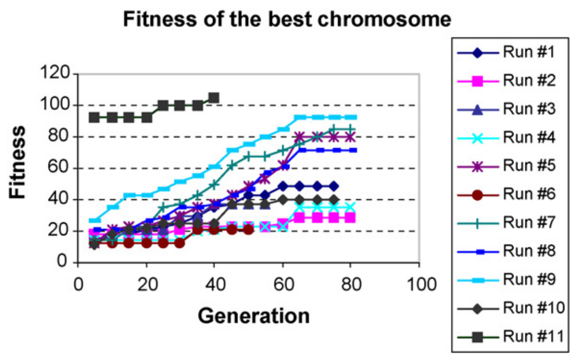

Fig. 7. The evolution of the fitness of the best chromosome during the discovery of the rule set for gene preGAD67. The fitness was calculated using the metric described in Section 2.2, and the number 100 corresponds approximately to an average error of 1% in the forecasting for each time point. Since the evaluation function is linearly inversely proportional with the trajectory fitting error, an evaluation two times smaller indicates an error two times bigger and so on. Notice that the 11th run is seeded with the best individuals from each run, so its fitness starts at the same point where the top evaluation from the previous runs has stopped (the best chromosome from the 10th run). Some of the curves stop before the end of the 80th generation because one of the other stopping criteria has been met.

In four of the 11 runs performed, the stagnation criteria stopped the execution before the algorithm could finish the 80 generations. In the 11th run, the quality criterion was met before any of the other two criteria, stopping the algorithm halfway through the execution.

Competing operators were used. Crossover received an initial probability of 0.95 while the mutation operator has a probability of 1-crossover fitness. The probability of the crossover was linearly descending down to a minimum of 0.2 in the last generation. When chosen, the mutation operator would mutate each tree branch with a probability of 5%.

Each population had 100 individuals and the population module used elitism in order to preserve the best solutions, transferring the best two chromosomes with the highest evaluation function in the current generation to the next, while replacing the remaining 98 with newly generated individuals.

The number of different individuals in each population and the number of generations may be considered low for the genetic programming standards, but, as explained in the following section, these numbers allowed for an average error of 1% when calculating the expression values according to the rule base found, which is a good result by any standard.

The fact that good results were achieved in such a small number of generations may be a consequence of the fact that the crossover operator successfully preserves sub-expressions and therefore does not cause a large impact when changing in the GP tree or may be a consequence of specific problem characteristics. Therefore, when applying the proposed algorithm to new problems, users should be aware that larger number of individuals per population and a larger number of generations may be necessary.

In both problems described in this section the algorithm proposed here was run 10 times. The two best results from each run were used to seed the population for the 11th run, the results of which are reported here. This means that the initial population of the last run contained 20 chromosomes generated in previous runs and 80 randomly initialized individuals.

The rationale behind this approach is to try to mix the best performing chromosomes in every run in order to combine their schemes, instead of simply initializing randomly another population. This resulted in a individual with a higher fitness, as determined by the evaluation function used.

The fact that during the 11th run the best chromosome's evaluation improved does not necessarily imply that the previous runs have not stagnated. The ten previous runs added twenty different chromosomes to the new population, which was augmented by eighty randomly created chromosomes in order to create a new population with a good variability, but this does not permit any inference on the variability of the final populations of the previous runs.

The results shown in this paper are those from a division of the expression space of each feature into three different fuzzy sets. Other numbers of fuzzy sets were tried (2, 5, 7, 11 and 15 different sets), but no significant improvement in the results was obtained. For each feature under evaluation an average of two rules per fuzzy set would be allowed, which would create up to six fuzzy rules per feature.

A rule simplifier was created, based on the regular expression mechanism of Perl. This simplifier reduces tautologies such as A AND A to their simpler form (in this case, A). The rules shown here have already gone through this simplification process.

3.2. Results

In all the results shown in this section, the rule set selected was the one with the highest evaluation in the last run. This rule set produced the best fit for the trajectories available, meaning that its prediction made the smallest error from all the chromosomes in the population.

The first test dataset used was generated by microarray experiments. The algorithm was applied to search for regulation strategies for a specific set of genes present in the reaction to cold of *Arabidopsis thaliana* and these results were compared against previous biological knowledge (Gilmour et al., 1998). *A. thaliana*, like many plants, increases its freezing tolerance when exposed to low nonfreezing temperatures. This process of cold acclimation is a multigenic and quantitative trait that is associated with complex physiological and biochemical changes (Hannah et al., 2005).

The dataset was obtained from The Arabidopsis Internet Research project (TAIR).

It is obvious that established statistical methods such as covariance analysis are not appropriate in this case, given the fact that there are close to 8000 genes and only seven data points in our data files. This is a typical situation where usual methods are unsuitable to extract meaningful relationships from the data, given the small number of time points. In such a situation, the application of the algorithm proposed presents itself as an interesting alternative.

The genes that are hypothesized to be responsible for the cold response are 16062_s_at, 17520_s_at and 16111_f_at. The genes 15611_s_at, 15997_s_at, 13018_at and 13785_at are some of the genes regulated by the above named ones.

For the last two elements in this short list, a rule database was discovered, using the technique described in the previous sections. No prior knowledge was embedded in the algorithm and no constraint on the results was applied to privilege the known regulators. The best solution found by the algorithm presented an average MAPE below 1.5% at each time point and a correlation of 0.95 with the real trajectory for the expression values of gene 13875_at. The rules obtained for gene 13875_at are the following:

(a) IF AND Lowly_Expressed(17413_s_at) NOT Highly_Expressed(16111_f_at) THEN Lowly_Expressed(13785_at)
(b) IF NOT Medium_Expressed (15714_at) THEN Lowly_Expressed(13785_at)
(c) IF NOT Medium_Expressed (17834_at) THEN Lowly_Expressed(13785_at)
(d) IF Lowly_Expressed (17421_s_at) THEN Medium_Expressed(13785_at)
(e) IF Medium_Expressed (16062_s_at) THEN Medium_Expressed(13785_at)
(f) IF OR Lowly_Expressed(17050_s_at) Highly_Expressed(16062_s_at) THEN Highly_Expressed(13785_at)

(g) IF Medium_Expressed (17034_s_at) THEN
Highly_Expressed(13785_at)
(h) IF NOT OR Highly_Expressed(16062_s_at)
Lowly_Expressed(15140_s_at) THEN
Highly_Expressed(13785_at)

Rules (a), (e) and (f) show a relationship with the known regulators, while rule (g) shows a candidate regulator (17034_s_at) that was considered interesting enough by the biologists who provided the data to warrant further investigation in the near future. From the other rules we can see the presence of 15140_s_at and 17050_s_at that show high correlation with 17520_s_at, a known regulator not present in the rules discovered. This may be due to the small number of data points available, but no further claims can be made about these genes.

In the second case, some prior knowledge was included and the program was asked to include necessarily activation from 17520_s_at and preferentially an inhibition from 16111_s_at. The best solution found by the algorithm presented an average MAPE below 1% at each time point and a correlation of 0.97 with the real trajectory for the expression values of gene 13018_at. The rules obtained were the following:

(a) IF OR Lowly_Expressed(17520_s_at)
Highly_Expressed(20351_at) THEN Lowly_Expressed(13018_at)
(b) IF NOT OR Medium_Expressed(14969_at)
Lowly_Expressed(17034_s_at) THEN Lowly_Expressed(13018_at)
(c) IF Medium_Expressed (13018_at) THEN
Lowly_Medium(13018_at)
(d) IF AND Lowly_Expressed (17421_s_at)
Highly_Expressed(16218_s_at) THEN
Medium_Expressed(13018_at)
(e) IF AND Medium_Expressed (16111_s_at)
Medium_Expressed(17200_at) THEN
Medium_Expressed(13018_at)
(f) IF AND Lowly_Expressed(17034_s_at) NOT
Highly_Expressed(14832_at) THEN Highly_Expressed(13018_at)
(g) IF Highly_Expressed (17520_s_at) THEN
Highly_Expressed(13018_at)

The results show that the relationship required was present in two different rules: (a) and (g). The desirable rule was also present, in rule (e). Another interesting feature is the presence of gene 17034_s_at, which was also deemed interesting in the previous set of rules.

The rules present a few genes that don't seem to "belong" in terms of previous knowledge. This kind of spurious control relationship will always be present in any method and is a consequence of the blessing of dimensionality previously mentioned (não foi mencionado no artigo, só na resposta aos revisores, acho!!). Even with the pre-processing methods described in Section 2.1, there are many possible solutions that can be extracted from the data, so spurious relationships could be present in any final solution.

If any of the spurious regulations are found to be unacceptable for biological reasons, a new run can be made, seeding the new population with the previous results and instructing the algorithm that some specific genes are forbidden. New solutions that do not include the unacceptable genes can thus be found. The possibility of including this knowledge is, as described above, a strong point of the proposed algorithm.

The second dataset was that of the expression of the genetic network of the central nervous system of a rat during its embrionary and newborn stage, specifically, glutamic acid decarboxylase (GAD), which is the enzyme responsible for the conversion of glutamic acid to gamma-aminobutyric acid (GABA), the major inhibitory transmitter in higher brain regions, and putative paracrine hormone in pancreatic islets. Two molecular forms of GAD (65 and 67 kDa, 64% amino acid identity between forms) are highly conserved and both forms are expressed in the CNS, pancreatic islet cells, testes, oviduct and ovary.

The dataset consisted of 112 genes measured during nine different stages of development, five of them embrionary, three of them during infancy and one of the adult rat.

The gene interaction diagram can be seen in (D'Haeseleer et al., 2000). The algorithm proposed was applied to the determination of the regulation for pre-GAD67, which is highly connected (five inputs, as hypothesized in this article). The rules obtained for Pre-GAD67 were the following:

(a) IF AND Lowly_Expressed(5HT2)
Medium_Expressed(cellubrevin) THEN
Lowly_Expressed(pre-GAD67)
(b) IF Highly_Expressed (GAD65) THEN
Lowly_Expressed(pre-GAD67)
(c) IF Lowly_Expressed (pre-GAD67) THEN
Lowly_Expressed(pre-GAD67)
(d) IF Lowly_Expressed (GRg3) THEN
Lowly_Expressed(pre-GAD67)
(e) IF Medium_Expressed (GRa3) THEN
Medium_Expressed(pre-GAD67)
(f) IF Highly_Expressed (GAD65) THEN
Medium_Expressed(pre-GAD67)
(g) IF Lowly_Expressed (GRb3) THEN
Highly_Expressed(pre-GAD67)
(h) IF Lowly_Expressed (GRa4) THEN
Highly_Expressed(pre-GAD67)
(i) IF Lowly_Expressed (GRa3) THEN
Highly_Expressed(pre-GAD67)

Some features of these rules are as follows:

1. GAD65, which is a supposed repressor, indeed appears as such, in two rules: (b), where its high expression level leads to a low expression level of pre-GAD67 and (f), which shows a mild repression stating that if it is highly expressed, then a medium expression level of pre-GAD67 results. Note that the presence of rule (b) does not allow the alternative interpretation as a mild activation.
2. GRa3 is also a supposed repressor, which is also shown in the same way as in the previous item, by rules (e) and (i).
3. Pre-GAD67 has a feedback activation mechanism. This could be shown by rule (c), but this may be an extreme extrapolation, since it is obvious that a low expression of a gene correlates well with another low expression of itself, if no other conditions are given. Nevertheless, it is satisfying to see the relationship produced by the algorithm.
4. Rule (d) shows another finding about a supposed relationship. GRg3 is an activator of pre-GAD67 and that is correctly expressed by this rule, which states that a low expression level of the activator will lead to a low expression level of the activated gene. Of course, this is not true in all activator–activatee relationships, because there are exponential, enabler and sigmoid relationships in nature that are not captured by this kind of rule.

The algorithm was also applied to another gene in the set, GAD65. This gene is much less connected than preGAD67, having only two inputs. Nevertheless, the number of rules suggested to initialize the algorithm was the same as in the case of preGAD67. This was done in order not to influence the results and to let the algorithm try to prune spurious relationships by itself. The rules obtained for GAD65 were the following:

(a) IF Lowly_Expressed(GAD65) THEN Lowly_Expressed(GAD65)
(b) IF Highly_Expressed (preGAD67) THEN Lowly_Expressed(GAD65)
(c) IF Highly_Expressed (GRb3) THEN Lowly_Expressed(GAD65)
(d) IF AND Highly_Expressed (GRg3) Highly_Expressed (G67I86) THEN Medium_Expressed(GAD65)
(e) IF Lowly_Expressed (GRa3) THEN Medium_Expressed(GAD65)
(f) IF Highly_Expressed (GAD65) THEN Highly_Expressed(GAD65)

These rules have some characteristics worth emphasizing:

1. According to previous work, GAD65 self activates. This can be seen in rules (a) and (f). To be perfectly honest, however, it should be pointed out that, as in the previous example, every gene will have a high self-correlation, so this discovery should be taken with a bit of caution.
2. According to previous work, GRb3 and preGAD67 are inhibited by GAD65. Rules (b) and (c) describe the mechanism backwards, although they capture the essence of the synchronized gene expression changes.
3. GRg3 was predicted as an inhibitor previous work (D'Haeseleer et al., 2000), and its mild inhibitory activity is shown in rule (d). It is necessary to point out that, according to previous work, G67I86 is upregulated by GRg3, so the AND rule may be considered to have, in essence, discovered a stronger relationship between GRg3 and GAD65.

When compared with previous work done on this set of data (D'Haeseleer et al., 2000), it can be said that the results obtained by the algorithm here proposed are quite similar to the ones obtained in previously published work and accepted as biologically plausible.

The work by (D'Haeseleer et al., 2000) predicts that the inhibitors for gene preGAD67, are GRa3, GRa2 and GAD65, while the activators are GRg3 and the gene preGAD67 itself. In the results given by the algorithm proposed here, there are four of the previously predicted regulators (GRa3, the preGAD67 itself, GRg3 and GAD65), all of them with the correct role in the gene dynamics.

In the case of GAD65, (D'Haeseleer et al., 2000) predicts that there is only one activator (GAD65 itself) and one repressor (GRg3). The two previously predicted regulators are present in the best rule set discovered, as well as several real relationships that occur in the genetic network.

It is important to point out that since data is scarce and the temporal analysis is not accurate, some cause-effect relationships may be reversed by the algorithm, as seen in rules (b) and (c) for GAD65. Nevertheless, it is quite remarkable that such a relationship was found at all.

The modeling ability of the algorithm proposed can be seen in Fig. 8, where the real and calculated trajectories for preGAD67 and GAD65 are compared. A comparison is made between the expression level predicted by the algorithm and the actual expression level present in the data and each point in the graph represents the expression levels, either real or calculated, depending on the curve considered, for the genes under analysis.

Notice that although there are some minor differences in the absolute values predicted by the algorithm, the

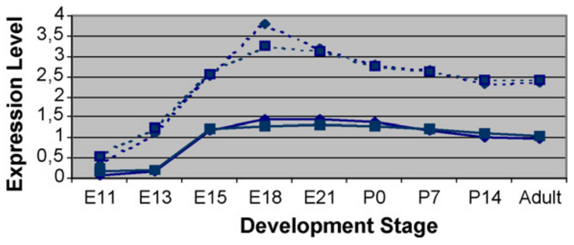

Fig. 8. Calculated (squares) and real (lozenges) trajectories for pre-GAD67 (solid lines) and GAD 65 (dotted lines) in the rat central nervous systems in early stages of development for the chromosome with the highest evaluation generated by our algorithm. The lines represent the predicted and actual expression levels for each one of the genes. The *x*-axis corresponds to different stages of development (the first five from embryonic days 11–21, the next 3, postnatal days 0–14, and the last one, at adult stage), while the *y*-axis corresponds to expression level (measured as fold change to a basis level). Notice that the curve shapes are almost the same.

curve shape is almost exactly the same. This suggests that the regulation process underlying the gene expression dynamics may have been discovered correctly.

In both cases, along with the known regulations, there are several candidate regulators that are probably spurious. This is to be expected, due to the small number of data points available. Nevertheless, the point of using the algorithm proposed has been made, since the original goal was to restrict the search in the biological workbench.

A researcher who receives the set of rules obtained for gene 13875_at, for instance, would have a short list of nine genes (out of 8000) that are considered suitable candidates by the algorithm proposed here. This means that the search space for a regulator has been reduced by a factor of 99.9%, simply by applying the proposed computer algorithm. This was replicated in both experiments described in this paper and therefore it is reasonable to conclude that the results are not merely due to chance correlations and therefore the algorithm may be considered for further use by biological researchers.

4. Conclusions and further work

The algorithm proposed shows results that are interesting when dealing with scarce data. When applied to known regulation models, the results generated are very similar to known results in biology. This suggests that the algorithm may be a tool that will help uncover other regulation processes that are still not known and also that in the future the algorithm could be applied to genes whose regulation is unknown so that the hypotheses generated could be tested in a lab.

The results were presented using fuzzy rules, which use linguistic terms and are, therefore, similar to the way a person naturally understands complex phenomena. This makes the results more readable and understandable, which compares favorably with other complex data mining algorithms, such as support vector machines, neural networks and logistic regression, which usually produce models that are not easily interpretable by biologists and biomedical researchers, given the high number of variables and parameters (Vinterbo et al., 2005).

Using fuzzy logic makes it easier for a hypothesis to be tested. Since the algorithm allows the user to insert any rules he may deem interesting, the algorithm can become a test bed for inexpensively discarding non-working regulation strategies. Therefore, a biologist will save time and money, avoiding unnecessary lab experiments. In a way, the tool can complement biological intuition, allowing biologists to use their insight and then test their hypotheses quickly and with no further costs.

These two characteristics are the two major strong points in this application. The idea of generating a small set of models that represent possible interactions that can help focus hypothesis generation for biological experiments (i.e., reducing the search space for gene knockout experiments) has already been proposed, for instance in (Repsilber et al., 2002; Sokhansanj et al., 2004) and other contemporary works in this field. Nevertheless, although these papers use the descriptive power of fuzzy logic, they do not possess simple mechanisms for the incorporation of prior biological knowledge, fundamental in order to guide the algorithm.

The ability to generate hypotheses leads to a further use of the algorithm proposed. There are many papers, such as (Ideker et al., 2000) and (Wagner, 2001) that propose the inference of genetic networks using perturbation analysis. The idea behind such methods is that additional information about a genetic network may be gleaned experimentally by applying a directed perturbation to the network, and observing the steady-state expression levels of every gene in the network in the presence of the perturbation. The problem with this approach is that perturbation experiments may become expensive, especially if they are repeated in order to eliminate spurious results. Thus the algorithm proposed here could be used to choose the genes to be deleted or over-expressed, so that less experiments are performed and money is saved.

It is very difficult, and probably even impossible, to find the real set of regulators with such a scarce dataset, but from the results described in this paper, it is possible to conclude that the algorithm proposed can help narrow the search making it more cost effective. When dealing with problems that have enough data points, it is strongly recommended that statistical or other well-established methods be used.

It is important to bear in mind that since the conclusions attained are based on a small amount of data that is not statistically sufficient, it is possible that spurious results will be obtained, especially false positives. With very small numbers of samples this will be a problem for any method, but with an increasing number of trajectories, the effects of chance will be minimized.

Nevertheless, it is important to emphasize that the results obtained point to an effective discovery of regulation, since an elimination of up to 99.9% of the non-regulating genes was achieved and the ones remaining included the genes searched for; that is, the genes that are known to be involved in *A. thaliana* cold response.

In this paper, 'validation' of the rules was done mostly on the basis that the results obtained with their use relate well to prior knowledge. Of course, the best validation of any results would be to take them to the biological workbench for further testing. In fact, the rule bases obtained by the method here proposed should be treated as interesting candidates for the modeling problem, not as final results. Thus further testing, especially in the biological workbench, will always be required.

It is important also to understand that simply analyzing the data is not the final answer for the genetic network reverse engineering problem. It is also necessary to embed the available state of the art knowledge in the biological field in order to make any model obtained more complete (Schrager et al., 2002). The algorithm proposed here allows for the introduction of knowledge previously available in order to constrain the search space and obtain results closer to reality.

It is always advisable to use as much of the available knowledge as possible and in order to do that, we are currently studying the binding domain determination so that we can define those elements that are the most probable candidates for regulation and search exclusively among them.

References

Arnone, M.I., Davidson, E.H., 1997. The hardwiring of development: organization and function of genomic regulatory systems. Development 124, 1851–1864.

Bentley, P., 1999. Evolving fuzzy detective: an investigation in the evolution of fuzzy rules. In: International Conference on Computational Intelligence for Modelling, Control and Automation, vol. 1. IOS Press, pp. 17–19.

Carse, B., Fogarty, T.C., Munro, A., 1996. Evolving fuzzy rule based controllers using genetic algorithms. Fuzzy Sets and Systems, vol. 80. Elsevier Science, pp. 273–293.

Creighton, C., Hanash, S., 2003. Mining gene expression databases for association rules. Bioinformatics 19 (1), 79–86.

Dasgupta, D., Gomes, F.A., 2001. Evolving complex fuzzy classifier rules using a linear tree genetic representation. In: Proceedings of GECCO'2001: Genetic Evolutionary Computation Conference. Morgan Kauffman Publishers, pp. 540–551.

Dasgupta, D., Gomez, J., 2002. Evolving fuzzy classifiers for intrusion detection. In: Proceedings of the 2002 IEEE Workshop on Information Assurance, US Military Academy.

De Jong, K., Spears, W., 1991. Learning concept classification rules using genetic algorithms. In: Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, pp. 651–656.

D'Haeseleer, P., Liang, S., Somogyi, R., 2000. Genetic network inference—from co-expression clustering to reverse engineering. Bioinformatics, vol. 16, no. 8. Oxford University Press, pp. 707–726.

Dounias, G., Tsakonas, A., Jantzen, J., et al., 2002. Genetic programming for the generation of crisp and fuzzy rule bases in classification and diagnosis of medical data. In: Proceedings of the Neuro-Fuzzy Conference. NAISO Organizer, pp. 92–98.

Duggan, D.J., Bittner, M., Chen, Y., et al., 1999. Expression profiling using cDNA microarrays. Nature genetics supplement, vol. 21, pp. 10–14.

Fogel, G.B., Corne, D.W., 2003. Evolutionary Computation in Bioinformatics. Morgan Kaufmann Publishers.

Freitas, A.A., 2003. A survey of evolutionary algorithms for data mining and knowledge discovery. In: Ghosh, A., Tsutsui, S. (Eds.), Advances in Evolutionary Computation. Springer-Verlag, New York, pp. 819–845.

Gilmour, S.J., Zarka, D.J., Thomashow, M.J., et al., 1998. Low temperature regulation of the Arabidopsis CBF family of AP2 transcriptional activators as an early step in cold-induced COR gene expression. Plant J. 16 (4), 433–439.

Hannah, M.A., Heyer, A.G., Hincha, D.K., 2005. A global survey of gene regulation during cold acclimation in *Arabidopis thaliana*. PLoS Genet. 1 (2), 26–43.

Hiirsalmi, M., Kotsakis, E., Pesonen, A., Wolski, A., 2000. Discovery of fuzzy models from observation data. Research Report, Research Report TTE1-2000-43m VTT Information Technology, available at http://www.geocities.com/ekotsakis/EN/publications/reports/rr0043-discovery.pdf.

Ideker, T.E., Thornsson, V., Karp, R.M., 2000. Discovery of regulatory interactions through perturbation: inference and experimental design. In: Pacific Symposium on Biocomputing, vol. 5, pp. 302–313.

Koza, J.R., 1992. Genetic programming. In: On the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge.

Linden, R., Bhaya, A., 2002a. Reverse engineering of genetic networks under the boolean networks model using variable-length genetic algorithms. In: Proceedings of the XIII Artificial Neural Networks in Engineering (ANNIE/2002), ASME Press, pp. 535–541.

Linden, R., Bhaya, A., 2002b. Reverse engineering of genetic networks under the boolean networks model using variable-length genetic algorithms. In: Anais do I Workshop Brasileiro de Bioinformática, pp. 89–91.

Linden, R., Bhaya, A., 2003. Extracting gene relationships from microarrays using fuzzy logic and genetic algorithms. In: Mondaini, R. (Ed.), Proceedings of the Third Brazilian Symposium on Mathematical and Computational Biology, pp. 352–378 (Editora E-Papers).

Mcadams, H.H., Shapiro, L., 1995. Circuit simulation of genetic networks. Science 269, 650–656.

Mitchell, M., 1996. An Introduction to Genetic Algorithms. The MIT Press, Cambridge, MA.

Moore, D.S., McCabe, G.P., 2005. Introduction to the Practice of Statistics, fifth ed. W.H. Freeman, New York.

Repsilber, D., Liljenstrom, H., Andersson, S.G., 2002. Reverse engineering of regulatory networks: simulation studies on a genetic algorithm approach for ranking hypotheses. BioSystems 66, 31–41.

Rutter, M.T., Zufall, R.A., 2004. Pathway Length and Evolutionary Constraint in Amino Acid Biosynthesis, vol. 58, no. 2. Springer-Verlag, New York, pp. 218–224.

Schena, M., Shalon, D., Davis, R.W., et al., 1995. Quantitative monitoring of gene expression patterns with a complementary DNA microarray. Science 270 (5235), 467–470.

Schrager, J., Langley, P., Pohoriller, A., 2002. Guiding revision of regulatory models with expression data. In: Proceedings of the Pacific Symposium on Biocomputing, pp. 486–497.

Smolen, P., Baxter, D., Byrne, J.D., 2000. Modeling transcriptional control in gene networks—methods, recent results, and future directions. Bull. Math. Biol. 62, 247–292.

Sokhansanj, B.A., Fitch, J.P., Quong, J.N., Quong, A.A., 2004. Linear fuzzy gene network models obtained from microarray data by exhaustive search. BMC Bioinformat. 5, 108.

Van Someren, E.P., Wessels, L.F.A., Reinders, M.J.T., 2000. Linear modeling of genetic networks from experimental data. In: Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology. AAAI press, pp. 355–366.

Súer, G.A., Badurdeen, F., Thangalevu, B., 2002. Capacitated lot sizing by using multi-chromosome crossover strategy. In: Dagli, C., et al. (Eds.), Intelligent Engineering systems through Artificial Neural Networks, vol. 12, pp. 281–286.

Thomas, R., 1998. Laws for the dynamics of regulatory networks. Int. J. Dev. Biol. 42, 479–485.

Venet, D., Maenhaut, C., Bersini, H., 2001. Modeling and determination of regulation of gene expression: the binary switch model. In: Proceedings of the Second International Conference on Systems Biology, USA, pp. 239–247.

Vinterbo, S.A., Kim, E., Ohno-Machado, L., 2005. Small, fuzzy and interpretable gene expression based classifiers. Bioinformatics 21 (9), 1964–1970.

Wagner, A., 2001. How to reconstruct a large genetic network from n gene perturbations in fewer than n^2 easy steps. Bioinformatics 17 (2), 1183–1197.

Wain-Hobson, S., Renoux-Elbé, C., Vartanian, J.P., et al., 2003. Network analysis of human and simian immunodeficiency virus sequence sets reveals massive recombination resulting in shorter pathways. J. Gen. Virol. 84, 885–895.

Weaver, D.C., Workman, C.T., Stormo, G.D., 1999. Modeling regulatory networks with weight matrices. In: Pacific Symposium on Biocomputing, pp. 112–123.

Yang, Z.R., Thomson, R., et al., 2003. Searching for discrimation rules in protease proteolytic cleavage activity using genetic programming with a min-max scoring function. BioSystems, vol. 72. Elsevier Ireland Ltd, pp. 159–176.

Zhou, C., Xhiao, W., Tirpak, T.M., Nelson, P.C., 2002. Evolving classification rules with gene expression programming. Motorola Labs.