Reverse Engineering of Genetic Networks Using Variable Length Genetic Algorithms with Boolean Networks

Ricardo Linden¹, Amit Bhaya²

¹FSMA-RJ/COPPE R. Campos Elísios S/Nº - Macaé-RJ
rlinden@pobox.com
²COPPE - Federal University of Rio de Janeiro (UFRJ)
Cidade Universitária CEP 21945-970 Rio de Janeiro, RJ Brazil
amit@nacad.ufrj.br

Abstract Nowadays, given the new microarray technology, a huge amount of data on gene expression is available. In order to understand the genetic expression process more completely, we need to know the control structure of the genetic expression. We present a new scheme for a genetic algorithm that shows promising results when modelling boolean networks. We can effectively and quickly obtain good approximations of the underlying control structures, given very little information on the possible trajectories.

1. Introduction

Nowadays, the data available on gene expression is growing exponentially, but in order to understand the underlying dynamics of the gene transcription regulation we need more than merely collecting large amounts of experimental data by gene expression assays. A framework for deriving and expressing the biochemical architecture of genetic systems, using experimental data, is required [4].

Most cells have the same DNA, but they differentiate because of many factors, including transcriptional regulation. This occurs through the combinatorial action of gene products on sequence elements close to each gene’s transcriptional start site, establishing the foundation for cell differentiation.

Nowadays we use Boolean networks to model gene networks. A Boolean network, denoted \( G(V,F) \), consists of a set \( V=\{v_1,v_2,\ldots,v_n\} \) of nodes representing genes and a list \( F=\{f_1,f_2,\ldots,f_n\} \) of Boolean functions, where a Boolean function \( f_i(v_{i1}, v_{i2}, \ldots, v_{ik}) \) with inputs from specified nodes \( v_{i1}, v_{i2}, \ldots, v_{ik} \) is assigned to each node \( v_i \). For a subset \( U \subseteq V \), an expression pattern of \( U \) is a function \( \psi \) from \( U \) to \{0; 1\}. An expression pattern of \( V \) is also called a state of a Boolean network, and represents the states of nodes (genes), where each node is assumed to take either 0 (not-expressed) or 1 (expressed) as its state value. The expression pattern at time \( t+1 \) is determined by Boolean functions \( F \) of the expression pattern at time \( t \) [1].

Even though boolean networks are extremely simple, they exhibit some behaviour analogous to the real development of organisms. Concepts such as cell types can be modelled as attractors in state space and differentiation as transition between attractors and stability of expression patterns as basins of attraction. Thus, it may be argued that using a simple model such as a boolean network, we can gain useful insight into the extremely complex issue of development.

There are some features of cell regulation that point to boolean behaviour: for example, DNA expression in which inhibition occurs through the binding of an element to an operon [2], and many elements are regulated by sigmoidal processes, which can be crudely approximated by a step function, which is essentially boolean. Other cellular features are essentially binary: for instance
In order to model the dynamics of transcriptional regulation, we use genetic algorithms (GA). The idea behind GAs is to mimic the natural evolution of the species in order to create a new kind of search technique. A GA is fully defined by the coding scheme (how each individual will be represented in the computer), the operators (both mutation and crossover) and the evaluation or fitness function (i.e., a measure of the quality of the current solutions (individuals) to the problem at hand).

The chromosome structure we used is depicted in figure 1. Inside the chromosome we have n genes, and each one of them stores the elements that regulates one node of the boolean network. The regulation consists of two parts: the nodes that affect the current node and the boolean function that their combination generates.

The mutation operator can change any position of the gene (either regulation element or boolean function bit) based on a determined probability. The crossover operator is very simple. We simply exchange the regulation strategies for each node of our boolean networks according to a dice throw. This is equivalent to the uniform crossover operator, but applied to the whole regulation of each node, not to the components of this regulation.

In order to evaluate the performance of the chromosome, we stored t trajectories with n steps each. Each trajectory represents the “real” behaviour of the network we want to model. Therefore, every network receives the first state of each trajectory and the GA calculates the intermediate and final steps for this network. The number of bits that differ at each stage of the trajectory is added and averaged over the number of steps. Afterwards, each chromosome is penalized according to the number of non-zero bits (active relationships) it represents, so that the shorter chromosomes get extra credits.

Analyzing our results, we noticed that most runs found a perfect regulation for some nodes. Therefore, we combined the best results of our GA in order to improve our results. We selected the top 8 solutions, based only on the GA evaluation function, in order not to insert any unfair bias, and combined all possible regulations.

2. Results

In order to test our GA, we created a few examples that contain interesting features. It must be stressed that our implementation is efficient, and one run of the GA takes an average of 5-10 seconds in a desktop PC (Pentium MMX 166 MHz).

We presented the algorithm with some purely abstract examples that include interesting characteristics, like periodicity and the algorithm was able to identify all the correct relationships. Afterwards, we created trajectories that resemble the the lac operon in E. Coli and our algorithm discovered the correct relationships among elements, even if the boolean functions were not exact. Still, the technique worked remarkably well for such a simple model.
3. Comparison with previous work

Even though the matrix model described in [5] is continuous, it is computationally complex, the solutions found are limited on the number of inputs and it cannot model such ubiquitous functions such as a logical AND. Since our GA does not constrain the number of inputs, it can easily find more realistic regulations.

The comparison with the work described in [1] is more straightforward. It uses exactly the same binary model we do, but limits the number of inputs to two, allowing for an exhaustive search strategy. This limitation is not realistic since in nature most genes are regulated by 4 to 8 elements [3]. This limitation obviously does not apply to our work, since the GA admits any number of regulating nodes.

4. Conclusions and further work

Our GA works really well with small networks. It seems that the algorithm proposed does not scale well when presented bigger networks. Therefore, in order to apply the technique described in this paper we need to divide the genes in overlapping clusters so that we can extract many small networks from a big network and can apply the GA. Given that cis-regulatory control strategies are modular [3], separating them in overlapping clusters should not limit our ability to model them.

We need to improve the crossover operator in our GA. The current operator is quite crude and does not perform a fine tuning of the regulation network. In addition, we need to study the effects of the evaluation function. Penalizing the most complex relationships makes the networks tend to the simplest solution, but makes it more difficult to find complex correct solutions.

Notwithstanding the comments above, we obtained some very satisfying results. Our GA was able to discover many useful relationships and emulate the dynamics of the trajectories quite well. It is, of course, necessary to develop reliable statistical measures of the quality of the solution in fitting and extrapolating the data.

Another future direction is towards the application of this GA technique, perhaps hybridizing it with Fuzzy sets, to a continuous network. In order to have realistic simulations, we must work in the domain of real numbers.

5. References