Bio-inspired Reverse Engineering of Regulatory Networks

Cristina Costa Santini, Gunnar Tufte and Pauline Haddow

***Abstract*— Regulatory networks are complex networks. In this paper the challenge of modelling these complex networks is addressed. A modelling architecture based on bio-inspired techniques applied to the search for Boolean network representations of sought regulatory networks is presented. Two bio-inspired techniques are investigated when applied to the reverse engineering of a Boolean network model: a Genetic Algorithm and an indirect reverse engineering method. The latter is proposed as a means of addressing the challenge of modelling large and complex networks.**

I. Introduction

Modelling regulatory networks is an ongoing challenge in the field of systems biology due to the inherent complexity of such networks. Regulatory networks consist of both signal transduction networks and genetic transcription networks. Signal transduction networks transmit signals from the boundary of a cell to the nucleus in a combinatorial process and genetic transcription networks control if and how genes are transcribed.

The complexity inherent in signal transduction pathways arises, not only from the size i.e. the number of pathways and components, but from the crosstalk between pathways and from the convergence of pathways. Further, particular genes influence multiple functions (pleitropy), providing a further level of complexity within regulatory networks [1]. Such complexity was confirmed by the publication of the human genome in 2001. Much fewer genes than originally estimated were shown to be present thus illustrating the fact that genes must interact in even more complex ways in order to maintain homeostasis [2].

Any form of modelling of biological processes needs to address the abstraction level sought i.e. the level of detail to be included and the scope or aim of the model [3]. Further, modelling the dynamics inherent in complex biological processes requires vast resources. Although unlimited resources may be assumed, realistically effective computation platforms are required.

A further challenge inherent in modelling biological processes is the information available to modellers. On one side, there may be a vast amount of information or unstructured data, providing the modeller with a challenge to extract the relevant data. On the other side, data may not be complete thus providing a challenge as to how to cope with incomplete information.

In general, modelling approaches to regulatory networks may be said to attempt to provide a one to one description of the network or sub-network under investigation. As the complexity and the size of the network increases, challenges with respect to both the model creation and the resources required to run the dynamics of the model increase.

In this paper, an indirect reverse engineering method is proposed to address the challenges highlighted. It is investigated through the reverse-engineering of a simple model, the yeast cell-cycle network model [4].

This paper is organised as follows: Section II presents the motivation and the context of the work. Section III presents an overview of the modelling architecture. Section IV presents the search techniques applied and Section V presents the representation chosen for the regulatory networks: Boolean networks. In Section VI experiments and results of the reverse engineering of the yeast cell-cycle network model are presented and discussed. Section VII discusses the future work and Section VIII concludes the paper.

II. Motivation

Reverse engineering is an important first step in the modelling of biological regulatory networks, when, even though microarray data is available, there is no mechanistic understanding of the regulatory processes that are triggered by a certain stimulus.

Approaches to modelling regulatory networks have been classified by Huang [1] as local (detailed): aiming to achieve a detailed description of the processes modelled and global (abstract): aiming at understanding the general principles of the processes modelled. The former approach enables computational experiments to be executed, providing a faster and cheaper experimental platform than biological experiments. On the other hand, a systems understanding of regulatory networks will only be achieved if the models are more abstract i.e. the latter approach, so that biological principles can be extracted [5], [6]. A further aspect supporting the abstract approach is, as stated in Section I, the lack of available and suitable data. Quantitative measurements of biological components in space and time are still insufficient to provide a fully determined network.

There are several techniques being applied to modelling biological networks following both approaches, as reviewed in [7]. The more detailed, descriptive models can be seen as important 'bridges' between the hypotheses generated by the higher level abstract modelling techniques and the actual experimental validation. The approach required depends on the purpose of the model building and on the scope and abstraction level of the problem [3]. In this work, the abstract approach is chosen.

CRAB Lab, Department of Computer and Information Science (IDI), Norwegian University of Science and Technology (NTNU), Norway; emails: csantini@idi.ntnu.no, pauline@idi.ntnu.no and gunnar.tufte@idi.ntnu.no).

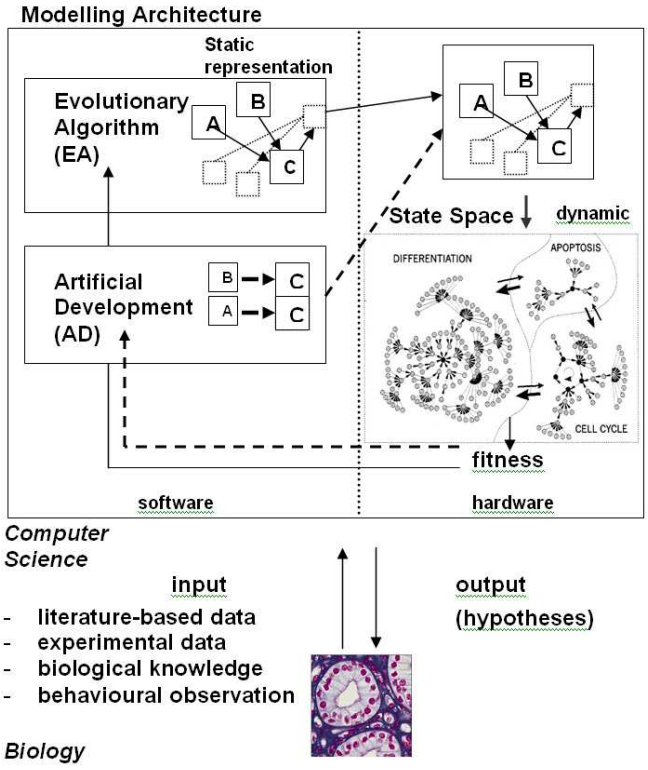

Fig. 1. Overview of the modelling architecture.

III. Overview of the Modelling Architecture

The goal is to achieve an understanding of the biological principles that govern regulatory networks. Figure 1 shows the modelling architecture. This extends the work of Haddow [8] who proposed a bio-inspired approach for searching for suitable regulatory network structures where the dynamics of such sought networks are studied on a highly parallel hardware platform. The goal being to generate realistic hypotheses by a circular process involving several model attempts, hypotheses generation and biological feedback.

There are a number of unique elements in the approach proposed by Haddow [8]. It suggests evolutionary algorithm techniques (see section IV-A) to search for network solutions and introduces a newer technique termed artificial development (see section IV-B) so as to scale up the search to large-scale network solutions. A further unique element is the computational platform itself. To meet the processing challenges of simulating the dynamics of large-scale networks, a highly parallel hardware platform is proposed.

The modelling architecture is refined herein and, in particular, Boolean networks are introduced as the representation of the networks sought and an indirect reverse engineering method is proposed and investigated through the reverse-engineering of a simple model, the yeast cell-cycle network model [4]. This is a small network, allowing this preliminary reverse engineering experiment to be implemented in software. Nevertheless, the final aim of the work is to reverse engineer large-scale networks that are going to be executed in hardware. Therefore, a criteria the representation to be chosen had to fulfill was to be suitable for hardware implementation. Being based on simple processing elements, Boolean networks are suitable for a hardware implementation (see Section VII).

In this work, the two different bio-inspired techniques, evolutionary algorithms and artificial development, are applied as a search engine to find possible Boolean regulatory networks (BRN) (see Section V). These are executed and the state space analysis provides, for the case study in hand, an evaluation of their dynamics, i.e. a 'fitness' measure. Such a process is a cyclic process, as shown in Figure 1, stopping when some search criteria is met. The output being a BRN hypothesis. Refinement of the hypotheses is again a cyclic process where biologists are involved to evaluate the hypotheses and provide revised input to the modelling process, as shown. The following sections provide more details about the elements of the modelling architecture.

IV. Bio-inspired Techniques

A. Evolutionary Algorithms (EA)

Taking inspiration from Darwin's theories of evolution, evolutionary algorithms (EAs) [9] aim to exploit the power of biological evolution as a computational search and/or optimisation tool. The artificial evolution process (evolutionary algorithms) is a dynamic process where at a given point in time one has what is termed in biology a generation: a population of individuals for the given species. Biologically, a generation change is not an event but a constantly unfolding process. However, in artificial evolution, a new generation is introduced through the application of reproduction and mutation techniques to selected individuals of the current generation. Further, unlike biological evolution there is a specified goal i.e. achieving a correct 'fit' solution to the problem in hand.

EAs are challenged to produce scalable solutions due to their resource-greedy nature [10]. One source of the scalability problems of EAs is the use of a one-to-one mapping from genotype to phenotype. That is, the solution generated by the EAs may be directly mapped to the candidate solution. This means that all properties of the phenotype will need to be directly encoded in the genotype. A large complex phenotype will, therefore, require a large complex genotype.

B. Artificial Development (AD)

Nature's way of handling complexity clearly points in the direction of a non one-to-one mapping from genotype to phenotype. Biological development is nature's way of coping with scaling. It provides a plan of the phenotype, describing how the organism is to be built rather than a blueprint containing explicit information about every detail of the proposed phenotype.

Unlike Evolutionary Algorithms there are no established methods for Artificial Development that may be applied in this work. Many efforts have been made in the field to test out different approaches to AD. However, little consensus

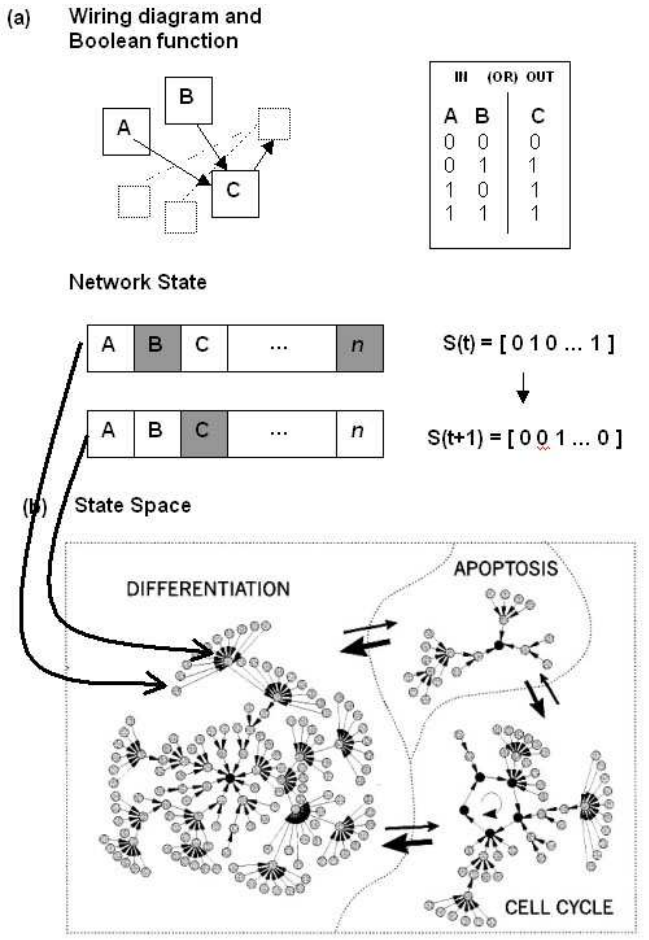

Fig. 2. (a) Basic structure of a Boolean regulatory network. (b) Scheme of the state space with the three attractors, illustrating the 'attractor landscape' (taken from [13]) in the notation proposed by [18].

has been achieved as to how such an approach should be applied and, in particular, little knowledge has been gained as to how different application areas should be approached. As such, a key challenge addressed herein is to find a way to apply Artificial Development to the reverse engineering of the Boolean network representation of regulatory networks.

Therefore, in order to address this challenge, a method is proposed and investigated, and is referred to as 'Artificial Development reverse engineering method'. In this method, the genotype is updated (or tuned) according to the phenotype's performance. More details about this method are given in Section VI-B, where we use the reverse engineering $in-silico$ experiments as an example.

V. Boolean Regulatory Networks

Boolean networks can be used as a representation for regulatory networks. A thorough discussion can be found on the seminal papers on the subject as well as in recent reviews and perspectives [11], [12], [13], [14], [15], [5], [1], [16], [17].

Boolean networks were first introduced by Kauffman [11]. In the Boolean idealization a regulatory network consists of a number *n* of elements representing genes/proteins, which can be either active 'on' (1) or inactive 'off' (0). The basic structure of a Boolean regulatory network is illustrated in Figure 2a. Each square represents an element of the network. These elements are interconnected as defined by the wiring diagram. Each element receives *k* inputs. A Boolean function is assigned to each element, dictating how the input is processed into an output. In the example shown the element C has two inputs (k=2) and its state is updated according to the OR Boolean function. The network state $S(t)$ is defined by the values of all the *n* elements of the network at a given time *t*. From the example, note that C=0 at time t and becomes 1 at time t+1 due to the values of A and B at time t. Their new values at t+1 reflect their own updates. By executing the network, i.e. executing the Boolean rules, a state transition occurs and all the states in the network are updated based on such transition rules such as the OR rule in the example.

The dynamics of this Boolean regulatory network will determine the various cell fates (attractor states of the network) such as differentiation, proliferation and apoptosis [13]. Figure 2b shows the scheme of the state space with the three attractors. Every grey dot in the state space represents a unique, transient (i.e., unstable) network state at a certain time (illustrated by the two external arrows on the left). These network states are connected by arrows that form the trajectories. The state space is divided (dotted line) into three basins of attraction with an attractor (black dots) at the centre of each. The size of the basins of attractions is correlated with the stability of the respective attractors. The arrows at the attractor boundaries and their size illustrate the probabilities of transition between the attractors.

Examples can be found in the literature that experimentally support this hypothesis of cell fates as attractor states. In addition, they provide further evidence that the Boolean network representation, even though abstract and simplistic, captures the essential aspects of regulatory networks. In [14], Huang and Ingber simulate the dynamics of the signalling system within capillary endothelial cells as a dynamic Boolean network and show that the attractors correspond to the cell's fates. In the work shown in [4], the authors propose and investigate a dynamic Boolean network model for the yeast cell-cycle (see Section VI for more details on this model). A further example of this approach is presented in [19], where the authors create a discrete Boolean model of signal transduction and they test the model's ability to replicate known qualitative behaviour. They support the idea that signal transduction networks are non-trivial decision making systems capable of clustering widely varying input combinations into a few cell states.

VI. Experiments and Results

In order to investigate the bio-inspired techniques when applied to the reverse engineering of regulatory networks, we performed preliminary $in-silico$ experiments aiming at the reverse engineering of the yeast cell-cycle network model, presented in [4] and reproduced in Figure 3. This model

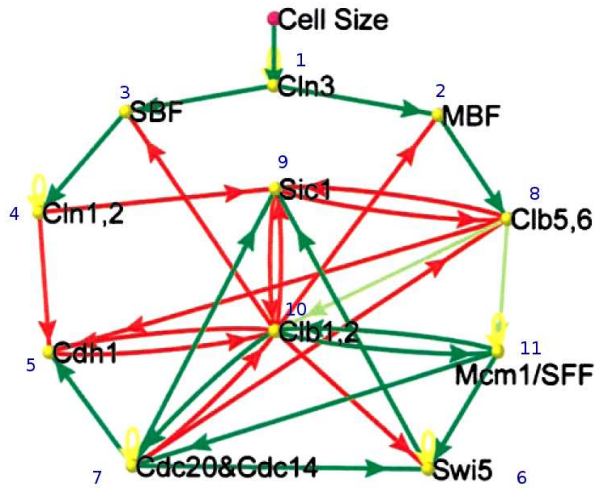

Fig. 3. Yeast cell-cycle network. From [4].

was chosen for being a simple BN model that, even though simple, models successfully the yeast cell-cycle phases and stationary states.

In the model, each node *i* has only two states, S_i=1 and S_i=0, representing the active and inactive state of the protein, respectively. The protein states in the next time step are determined by the protein states in the present time step via the following transition rule:

$$S_i(t+1) = \begin{cases} 1, & \sum_j a_{ij} S_j(t) > 0 \\ 0, & \sum_j a_{ij} S_j(t) < 0 \\ S_i(t), & \sum_j a_{ij} S_j(t) = 0 \end{cases} \quad (1)$$

For the results presented here a_{ij}=1 for a green arrow from protein *j* to protein *i* and a_{ij}=-1 for a red arrow from *j* to *i*. A green arrow represents positive regulation and a red arrow represents 'deactivation' (inhibition, repression, or degradation).

The model also has 'self-degradation' (yellow loops) on those nodes that are not negatively regulated by others. This self-degradation is modeled as a time-delayed interaction: if a protein with a self yellow arrow is active at time t and if its total input is zero at t+1, it will be degraded, i.e. $S_i(t+1)$=0. Using this dynamic model, the attractors of the network can be found; each of the 2^{11}=2048 initial states in the 11-node network eventually flow into one of seven stationary states (fixed points). Those are shown in Table 1, where each fixed point is represented in a row. The first column is the size of the basin of attraction for the fixed point; the other 11 columns show the protein states of the fixed point[1].

The aim of the experiments presented is to investigate the feasibility of the proposed AD reverse engineering method. Therefore, reverse engineering experiments when applying

[1]Please refer to [4] for more information about the model and its dynamic properties, as well as for the results on network perturbations. It can be mentioned that we reproduced and extended with success those experiments.

	1 Cln3	2 MBF	3 SBF	4 Cln1,2	5 Cdh1	6 Swi5	7 Cdc20	8 Clb5,6	9 Sic1	10 Clb1,2	11 Mcm1
input1	+12	+1	+1	+3	-4	+7	+10	+2	-4	+8	+8
input2	0	-10	-10	0	-8	-10	+11	-9	+7	-9	+10
input3	0	0	0	0	-10	+11	0	-7	-10	-5	0
input4	0	0	0	0	+7	0	0	0	+6	-7	0
input5	0	0	0	0	0	0	0	0	-8	+11	0

Fig. 4. Table showing the yeast cell-cycle network elements and their respective positive regulations or 'deactivations'.

this method are shown. In addition, in order to get an insight into the problem at hand, i.e. to investigate how hard the problem is, another bio-inspired method is also applied: a canonical Genetic Algorithm with standard parameters.

The data set for the $in-silico$ experiments include only partial data about the network under investigation. The choice of only including partial data is motivated by the fact that in reality, i.e. when investigating a biological system, it is unlikely that it will be possible to have inform about all the initial conditions and corresponding attractors. The full data set for the 11-node network consists of 2048 initial states. In the $in-silico$ experiments shown here, information about only 100 of the possible initial states is provided.

In addition, we show the results of an $in-silico$ experiment when applying the AD reverse engineering method and when the data set includes only a few trajectories (time series data). A relevant investigation to biological applications.

Because the GA is applied directly to the Boolean network representation of the yeast cell-cycle network presented in this Section, we first show the GA results, followed by the results for the AD reverse engineering method.

A. Applying EAs

The Evolutionary Algorithm we used is a canonical Genetic Algorithm (GA); it has a one-to-one genotype-phenotype mapping and its genotype is evolved.

Figure 4 shows the table for the Boolean network representation that corresponds to the wiring diagram of the yeast cell-cycle network shown in Figure 3. Every column represents a network element, or node, and lines 1 to 5 inform which are the inputs to that node, as well as if they are positive regulations or a 'deactivations'. This representation shown in Figure 4 is what is called here in this experiment *the genotype*.

In order to perform a reverse engineering investigation, we have to assume that some biological information about the regulatory network is known a priori, but not all. A possible way of doing that is by specifying an initial representation, i.e. an initial genotype. For the results shown in this section, the inputs to nodes 2 to 7 are searched for, whereas the inputs to the other nodes are considered known biological information.

The GA will then search for the inputs to the unknown nodes. Consider node 2 for example: during the GA search it can be determined that it is positively regulated by nodes 4 and 6 and negatively regulated by node 11. Then the fitness of this network is determined by executing it to all and each of the given initial conditions and measuring the hamming distance between the obtained final attractor and the yeast

TABLE I

THE FIXED POINTS OF THE CELL-CYCLE NETWORK. FROM [4]

Basin size	Cln3	MBF	SBF	Cln1,2	Cdh1	Swi5	Cdc20	Clb5,6	Sic1	Clb1,2	Mcm1
1,764	0	0	0	0	1	0	0	0	1	0	0
151	0	0	1	1	0	0	0	0	0	0	0
109	0	1	0	0	1	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	1	0	0
7	0	1	0	0	0	0	0	0	1	0	0
7	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	0	0

network's final attractor for that initial condition, which can only be one of the seven attractors shown in Table I.

The executed network, or the dynamic network, or in nonlinear dynamics terms, the state space, is what is called here in this experiment *the phenotype*.

The average hamming distance for all the given initial conditions is then the phenotype distance d_P, with the fitness being $F = e^{-d_P}$ [20].

1) EAs Results: The parameters of the *in-silico* experiment[2] are:

- generations: 1,000;
- initial population: 2,000;
- elitism: 1,000;
- mutation: 600;

The first generation has 2,000 randomly created network solutions. The following generations are formed by the 1,000 best networks of the previous generation, 600 mutated networks and 400 new randomly created networks. Mutation is performed on 600 randomly chosen elements of the previous generation, not on the 600 best ones. With those parameters, a maximum of 10^6 different proposed networks can be executed and evaluated in a run.

The results to 10 runs are graphically shown in Figure 5. The green series shows the maximum fitness given 100 initial conditions, with average fitness 0.90. The exact solution for this data (fitness 1) was found in 30% of the runs (runs 2, 5 and 9). The fitness of the networks found in each run when executed and evaluated for the whole state space, i.e. for the 2048 initial states, is shown in the blue series, and is on average 0.75. Also, on average, a solution was found around generation 800, i.e. after 800x1,000 different proposed networks have been executed.

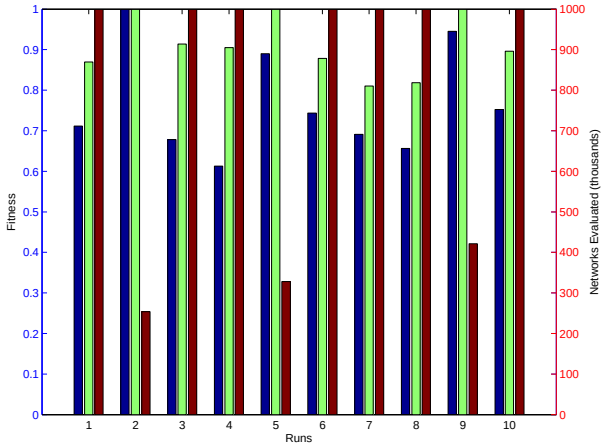

Fig. 5. Applying GA and 100 initial conditions. The green series shows the maximum fitness given 100 initial conditions and the blue series shows the fitness when the network found is evaluated for the whole state space, i.e. with 2048 initial conditions; the red series shows the number of networks evaluated, in each of the 10 runs.

B. Applying the Artificial Development (AD) reverse engineering method

We propose and investigate an AD reverse engineering method in which the biological network is represented by a list of influences, interactions or regulations. Here in this experiment this list is called *the genotype*. It carries the information about all the possible influences in the system, and informs the probability in which that influence is going to be present in the list of influences that uniquely represent the network, which for now on will be referred to as *the network*. For example, consider Figure 3, in which each arrow represents an influence, an activation or a deactivation: *'MBF (node 2) is positively regulated (green arrow) by Cln3 (node1)'* is one of the influences that compose the network.

Figure 6a explicitly shows the genotype used in the experiments shown here. The influence mentioned above, for example, is the first influence in the genotype.

As the reverse engineering investigation has to mimic a real biological situation, we suppose we only have some information about the network. In the experiment shown here, this is done by assigning different propabilities to the influences in the genome and by including some random influences that are not in the network. It can be observed that the first 29 influences in the genotype correspond to the interactions present in the yeast cell-cycle network (Figure 3). The other influences (columns 30 to 39) are included as possible influences. Consider the third row of the table in Figure 6a: the influences with probability 1 correspond to known information and will certainly be in the network and this probability will never change. The influences with probability less than 1 will have their probabilities updated according to the 'performance' of the network.

The adaptation process is depicted in Figure 7 and proceeds as follows: given the genotype, a network is built;

[2]This *in-silico* experiment is implemented in Matlab 7.6.0 and its execution time is in the hours time span.

(a) Initial genotype with the probabilities for each influence i

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
node	2	2	3	3	4	5	5	5	5	6	6	6	7	7	8	8	8	9	9	9	9	9	10	10	10	10	10	11	11	2	4	5	6	7	2	3	4	5	11
node	1	-10	1	-10	3	-4	-8	-10	7	7	-10	11	10	11	2	-9	-7	-4	7	-10	6	-8	8	-9	-5	-7	11	8	10	3	1	2	-8	-5	-9	-9	-9	6	6
prob_G	0.5	1	0.5	1	0.5	1	0.5	1	0.5	1	0.5	1	0.5	1	1	0.5	1	0.5	1	0.5	1	0.5	1	0.5	1	0.5	1	0.5	1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5

(b) A possible created network with the number of occurrences and with the normalized presence of each influence i

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	input
node	2	2	3	3	4	5	5	5	5	6	6	6	7	7	8	8	8	9	9	9	9	9	10	10	10	10	10	11	11	2	4	5	6	7	2	3	4	5	11	1
node	1	-10	1	-10	3	-4	-8	-10	7	7	-10	11	10	11	2	-9	-7	-4	7	-10	6	-8	8	-9	-5	-7	11	8	10	3	1	2	-8	-5	-9	-9	-9	6	6	12
# of occurencies	2	2	2	2	2	2	2	2	2	2	0	2	2	2	2	2	2	2	2	1	2	2	2	2	2	2	2	2	2	0	0	0	2	1	0	0	0	1	0	1
norm_presence	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1	1	0.5	1	1	1	1	1	1	1	1	1	0	0	0	1	0.5	0	0	0	0.5	0	

(c) Final genotype for run 5 with the updated probabilities for each influence i

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
node	2	2	3	3	4	5	5	5	5	6	6	6	7	7	8	8	8	9	9	9	9	9	10	10	10	10	10	11	11	2	4	5	6	7	2	3	4	5	11
node	1	-10	1	-10	3	-4	-8	-10	7	7	-10	11	10	11	2	-9	-7	-4	7	-10	6	-8	8	-9	-5	-7	11	8	10	3	1	2	-8	-5	-9	-9	-9	6	6
updated_prob_G	0.86	1	0.72	1	0.93	1	0.61	1	0.78	1	0.29	1	0.78	1	1	0.72	1	0.65	1	0.43	1	0.84	1	0.96	1	0.86	1	0.85	1	0.08	0.41	0.23	0.95	0.84	0.17	0.06	0.16	0.22	0.04

Fig. 6. Artificial Development reverse engineering method. (a) Table explicitly showing the initial genotype: nodes in row 1 are being (positively or negatively) influenced by the nodes in row 2; row 3 informs the probability with which each influence will be present in the network; (b) Table showing a possible created network: row 3 informs the number of times influence i appears in the network and row 4 is the normalization of row 3; (c) Table showing the final genotype for one of the runs: row 3 informs the final updated probabilities for each influence i.

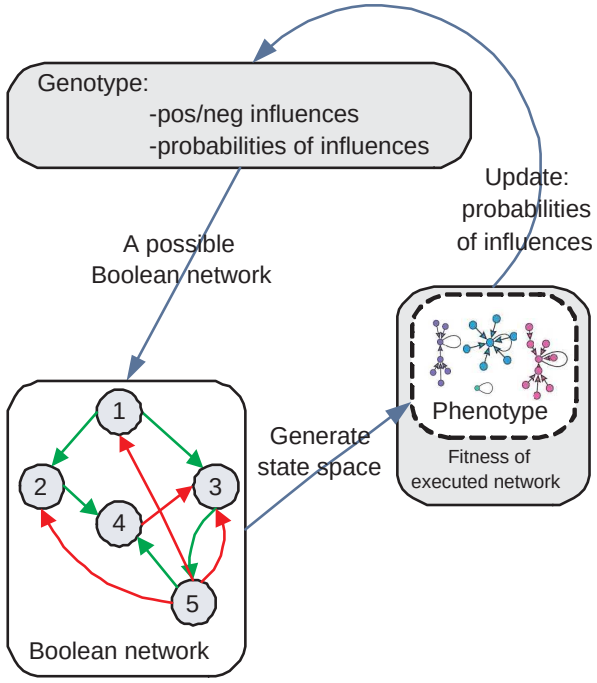

Fig. 7. The iterative process of the Artificial Development reverse engineering method proposed.

given the genotype shown in Figure 6a, a possible network is created. This is shown in Figure 6b. This process can follow diverse strategies. For the results shown here, all the genotype is checked twice, i.e. all the influences in the genotype have their respective probabilistic chances to be added to the network twice. The outcome is shown in the third row of the table in Figure 6b which informs how many times the respective influence is present in the network.

Then, just as with the experiments using GA, this network is executed according to (1) for each given initial condition. To execute the network means to investigate its dynamics for all the provided initial conditions. The result of this execution is the phenotype (in nonlinear dynamics terms, the state space). Following the method proposed here, *the phenotype* is therefore the product between *the genotype* (all possible influences and corresponding probabilities of being present in *the network*) and *the environment* (represented by the initial conditions).

The phenotype is evaluated by measuring its distance to the yeast cell-cycle network's phenotype. If this phenotype distance is smaller than all the phenotype distances previously measured, the genotype is updated. The update is a weighted update and it takes into account the current probability in the genotype ($prob\_G$ - Figure 6a third row) and the normalized list showing how many times a certain influence i appeared on the phenotype ($norm\_presence\_phenotype$ - Figure 6b forth row). For the results shown here the weight of the phenotype is 0.2. Therefore, the updated probability in the genotype for a certain influence *i* is $updated\_prob\_G_i = prob\_G_i * 0.8 + norm\_presence\_phenotype_i * 0.2$.

This process, illustrated in Figure 7, of creation and execution of the network and evaluation of the phenotype, followed by the possible update of the genotype, is repeated for a number of iterations or until a certain phenotype distance is achieved.

The idea behind this process is the following: a certain influence that is present in a 'good' phenotype should have an increased probability of being present in the network; in the same way, an influence that is not present in a good phenotype should have its probability of being in the network decreased.

Figure 6c shows the final genotype for one of the runs. Note that the list of interactions is the same as in the initial genotype (Figure 6a), but with the probabilities, shown in the third line, updated. After many updates, as expected, most of the probabilities up to influence 29 were updated towards 1, whereas the probabilities for most of the influences that are not in the original network (columns 30 to 39) were updated towards 0.

a) The experimental setup: In this experiment 100 randomly chosen initial conditions are given. Figure 8 shows the results for 10 runs. The target phenotype, which is the corresponding yeast cell-cycle network's final attractor for each of those 100 initial conditions, was found in all the 10 runs (green series). The fitness for the network found in each run when executed and evaluated for the whole state space,

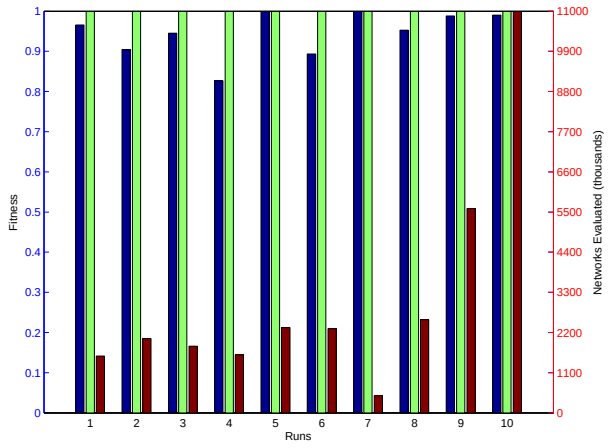

Fig. 8. Applying the AD reverse engineering method and 100 initial conditions. The green series shows the maximum fitness given 100 initial conditions and the blue series shows the maximum fitness when the network found is evaluated for the whole state space, i.e. with 2048 initial conditions; the red series shows the number of networks evaluated, in each of the 10 runs.

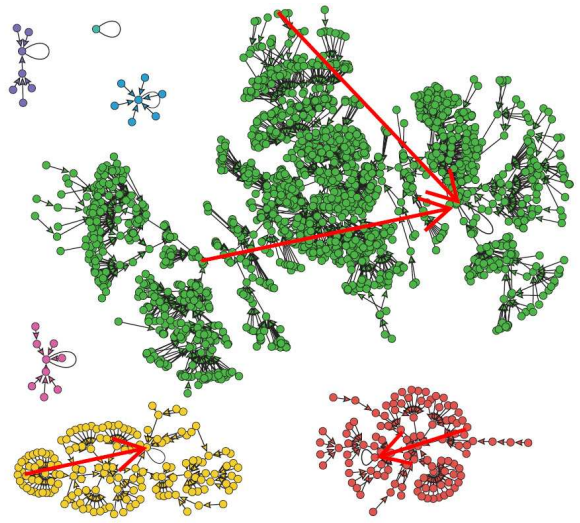

Fig. 9. State-space graph of the yeast cell-cycle network model, taken from [21]. Each node in the graph represents a state of the system, and an edge between nodes represent a dynamic transition between states, with a loop showing a fixed point. The seven fixed points of the network are shown in Table I. The arrows illustrate the 4 trajectories that were given to the reverse engineering algorithm.

i.e. for the 2048 initial states, is shown in the blue series and is on average 0.94. Most of the runs evaluated less than 3,000 proposed networks before the target was achieved (red series in Figure 8).

It is important to add that, when providing data about all the 2048 (and not only 100) initial conditions and corresponding final attractors, the network that corresponds exactly to the yeast cell-cycle network is found in all the 10 runs and in most of the runs less than 10,000 proposed networks were evaluated before the target was achieved.

b) Further Investigations: Motivated by the type of time series data[3] that is used by our collaborating group, the Gastrin Systems Biology group, another set of $in-silico$ experiments was performed. Instead of providing as data for the reverse engineering process the initial conditions and evaluating the corresponding final states, in these experiments the data is composed of only 4 trajectories, as graphically exemplified in Figure 9: two trajectories from the biggest basin of attraction and one from each of the two second biggest. In this situation, all the states in the four trajectories of the proposed networks are evaluated. In all the 10 runs, a network was found that would match exactly (fitness 1) the activation of all the nodes for the four given trajectories. Generalizing one of these solutions for the whole state space, i.e. for all the 2048 states, its fitness is 0.86. On average, 18,000 proposed networks were investigated in each run.

Another equivalent simulation was performed with 6 trajectories: the same four trajectories from before plus two from the biggest basin. Again, the networks proposed could match exactly all the trajectories, for all the 10 runs, and 14,000 proposed networks were investigated on average, in each run. In addition, when evaluating one of these solutions for the whole state space, i.e. for all the 2048 states, its fitness is 0.95. This result suggests that when more data is given, the method finds a solution that better represents the whole state space.

Both results show that the method is applicable to time series data.

C. Discussion

The search space of the Boolean network increases exponentially with the number of nodes: $(n^k)^n$. This implies that a sparsely connected network (k=2), with n=100 nodes, gives a total of 10^{400} possible networks.

In the experiments shown here the indirect method applied managed to generate valid networks by only evaluating a fraction of all the possible networks in the total search space. As such, an indirect method would be favorable if large-scale networks consisting of hundreds or thousands of nodes are to be the target.

Summarizing the results shown here, the indirect method, i.e. the Artificial Development reverse engineering method, investigated 3,000 networks and the best network had a fitness of 0.94. The Genetic Algorithm investigated 800,000 networks and the best network had a fitness of 0.75. Even though a direct comparison can't be made as neither method was exhaustively investigated in terms of their parameters, it can be noted that the number of networks investigated differ in two orders of magnitude. Therefore, the Artificial Development reverse engineering method shows promise but still needs to be investigated when applied to large-scale networks.

[3]Microarray mRNA data from 15 minutes to 24 hours after gastrin stimulation [22].

VII. Future Work

In order to investigate the AD reverse engineering method proposed here when applied to large-scale networks, an artificial large-scale network [23], [24] can be used. Artificial networks were previously used to investigate other reverse engineering methods. We plan at implementing this large-scale artificial network in hardware, as the dynamic execution of the network (or of some networks) in parallel will certainly represent a considerable speed-up of the reverse engineering process.

If a model of a Boolean network is to run on a sequential machine i.e. a standard microprocessor, as in the case of the results shown here, the run-time is given by the sequential update of the given number of nodes (n) in the network modelled. On the other hand, if the Boolean network is executed on a massive parallel hardware architecture, all nodes execute their Boolean function in parallel (one state change) and thus the run-time is not influenced by the number of nodes in the network. The simple processing element required by a Boolean network is only a simple Boolean function.

Reconfigurable technology is an existing technology providing a suitable architecture with large amounts of simple processing units. Further the technology is flexible, enabling both many reconfigurations, a requirement of the bio-inspired techniques, and dynamic reconfiguration, allowing for updates in the hardware during simulation of the dynamics of the modelled network.

VIII. Conclusion

This paper addresses the challenge of reverse engineering and modelling regulatory networks. The Boolean network representation is supported as a suitable abstract representation that even though simplistic, would correspond to the modelling purposes. An indirect reverse engineering method is proposed as a means of addressing the challenge of modelling large and complex networks.

A preliminary reverse engineering investigation is made through a set of $in - silico$ experiments, using two bio-inspired techniques: GA and the AD reverse engineering method proposed. Results show that both techniques are successful in reverse engineering the yeast cell-cycle network. However, on average, a better solution was found and less proposed networks were investigated when applying the AD reverse engineering method than when applying GA.

Acknowledgment

This work was carried out during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme. C.C.S. thanks Prof. Hidde de Jong's group for profitable discussions during the ERCIM Exchange Programme. We thank the Gastrin Systems Biology group at the Department of Cancer Research and Molecular Medicine, NTNU, for providing the biological realism.

References

[1] S. Huang, "Back to the biology in systems biology: What can we learn from biomolecular networks?" *Briefings in Functional Genomics and Proteomics*, vol. 2, no. 4, p. 279297, 2004.

[2] C. V. et al., "The sequence of the human genome," *Science*, vol. 16, pp. 1304–1351, 2001.

[3] H. Kitano, "Systems biology: A brief overview," *Science*, vol. 295, no. 5560, pp. 1662–1664, 2002.

[4] F. Li, T. Long, Y. Lu, Q. Ouyang, and C. Tang, "The yeast cell-cycle network is robustly designed," *PNAS*, vol. 101, no. 14, pp. 4781–4786, 2004.

[5] S. Huang, "Genomics, complexity and drug discovery: insights from boolean network models of cellular regulation," *Pharmacogenomics*, vol. 2, no. 3, pp. 203–222, 2001.

[6] P. K. Maini, "Making sense of complex phenomena in biology," in *In Silico Simulation of Biological Processes*, G. Bock and J. A. Goode, Eds. Novartis Foundation 2002, 2003, pp. 53–65.

[7] J. Fisher and T. A. Henzinger, "Executable cell biology," *Nature Biotechnology*, vol. 25, no. 11, pp. 1239–1249, 2007.

[8] P. Haddow, "Evolvable hardware : a tool for reverse engineering of biological systems," *ICES*, pp. –, 2008, to be published.

[9] J. Holland, *Adaption in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[10] S. Droste, T. Jansen, G. Rudolph, H. Schwefel, K. Tinnefeld, and I. Wegner, "Theory of evolutionary algorithems and genetic programming," in *Advancec in Computational Intelligence Theory and Practice*, H. P. Schwefel, I. Wegener, and K. Weinert, Eds. Springer, 2003, pp. 107–144.

[11] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed nets," *Journal of Theoretical Biology*, vol. 22, pp. 437–467, 1969.

[12] ——, *The Origins of Order: Self-Organization and Selection in Evolution*. Oxford University Press, 1993.

[13] S. Huang, "Gene expression profiling, genetic networks, and cellular states: an integrating concept for tumorigenesis and drug discovery," *Journal of Molecular Medicine*, vol. 77, no. 6, pp. 1432–1440, 1999.

[14] S. Huang and D. E. Ingber, "Shape-dependent control of cell growth, differentiation, and apoptosis: Switching between attractors in cell regulatory networks," *Experimental Cell Research*, vol. 261, pp. 91–103, 2000.

[15] S. Huang, "Cell state dynamics and tumorigenesis in boolean regulatory networks," in *Unifying Themes in Complex Systems*, A. A. Minai and Y. Bar-Yam, Eds. Springer Berlin Heidelberg, 2006, pp. 293–305.

[16] I. Shmulevich, E. R. Dougherty, and W. Zhang, "From boolean to probabilistic boolean networks as models of genetic regulatory networks," *Proceedings of the IEEE*, vol. 90, no. 11, pp. 1778–1792, 2002.

[17] S. Bornholdt, "Less is more in modeling large genetic networks," *Science*, vol. 310, pp. 449–451, 2005.

[18] A. Wuensche, "Genomic regulation modeled as a network with basins of attraction," in *Proceedings of the 1998 Pacific Symposium on Biocomputing*, vol. 4. World Scientific, 1998, pp. 89–102.

[19] T. Helikar, J. Konvalina, J. Heidel, and J. A. Rogers, "Emergent decision-making in biological signal transduction networks," *PNAS*, vol. 105, no. 6, pp. 1913–1918, 2008.

[20] P. Fernández and R. V. Solé, "Neutral fitness landscapes in signalling networks," *J R Soc Interface*, vol. 4, no. 12, pp. 41–47, 2007.

[21] K. Willadsen and J. Wiles, "Robustness and state-space structure of boolean gene regulatory models," *Journal of Theoretical Biology*, vol. 249, pp. 749–765, 2007.

[22] T. Bruland, E. Anderssen, K. Misund, T. S. Steigedal, B. Doseth, C. Fjeldbo, M. Langaas, H. Bergum, A. Lgreid, and L. Thommesen, "Basic common architectural motifs underlying gastrin induced gene expression can be identified by dimension reduction methods." 2008, manuscript in preparation.

[23] P. Mendes, W. Sha1, and K. Ye, "Artificial gene networks for objective comparison of analysis algorithms," *Bioinformatics*, vol. 19, no. 2, pp. 122–129, 2003.

[24] T. Van den Bulcke, K. Van Leemput, B. Naudts, P. van Remortel, H. Ma, A. Verschoren, B. De Moor, and K. Marchal, "Syntren: a generator of synthetic gene expression data for design and analysis of structure learning algorithms," *BMC Bioinformatics*, vol. 7, no. 1, p. 43, 2006.