Reverse engineering of gene regulation models from multi-condition experiments

Noel Kennedy, Alexandru Mizeranschi,
Paul Thompson, Huiru Zheng, Werner Dubitzky
University of Ulster, Northern Ireland, UK

***Abstract*—Reverse-engineering of quantitative, dynamic gene-regulatory network (GRN) models from time-series gene expression data is becoming important as such data are increasingly generated for research and other purposes. A key problem in the reverse-engineering process is the under-determined nature of these data. Because of this, the reverse-engineered GRN models often lack robustness and perform poorly when used to simulate system responses to new conditions. In this study, we present a novel method capable of inferring robust GRN models from multi-condition GRN experiments. This study uses two important computational intelligence methods: artificial neural networks and particle swarm optimization.**

Keywords—Gene regulatory networks; reverse-engineering; machine learning; multi-model fusion; optimization

I. Introduction

Regulation of gene expression (or gene regulation) refers to processes that cells use to create functional gene products (RNA, proteins) from the information stored in genes (DNA). Gene regulation is essential for life as it increases the versatility and adaptability of an organism by allowing it to express protein when needed. While aspects of gene regulation are well understood, many open research questions still remain [1]. The dynamic behavior and regulatory interactions of genes can be revealed by time-series experiments, that is, experiments that measure the expression of multiple genes over time [2]. In contrast to *static* gene expression data, the modeling and simulation approach allows the determination of stable states in response to a condition or stimulus as well as the identification of pathways and networks that are activated in the process [3]. A range of mathematical methods facilitating the reverse-engineering of quantitative, dynamic gene-regulatory network (GRN) models from time-series gene expression data have been reported in the literature [4]. Typical methods based on differential equations include the S-system (SS), artificial neural networks (ANN), and general rate law of transcription (GRLOT) method [5] [6].

One of the issues in reverse-engineering GRN models is the under-determined nature of the problem [7]. Essentially, this means that for the given data and the differential equations specifying the model, there is no unique solution to these equations. A consequence of this is that models derived from such data lack robustness. Thus, the predictive accuracy on unseen data sets is often poor. Various approaches have been employed to address this issue [8] [9]. The fact that the data is normally noisy and that the reverse-engineering process involves a non-deterministic element (optimization) is also a factor that influences robustness, but it is not as fundamental as the lack of complete information.

In the present study, we investigate the reverse engineering of robust GRN models based on repeated measures from the same GRN system under different conditions. We refer to experiments that generate data in this way as *multi-condition experiments*. The principal idea is that when the same GRN system is subject to different (non-destructive) conditions or stimuli, it will display a range of responses that together are more characteristic for the underlying system properties than a single response to a single stimulus. This concept is similar to data fusion, which is a process that integrates multiple sources of information representing the same real-world entity into a consistent and accurate model of that entity [10]. However, whereas in conventional data fusion the merging of information is normally achieved on the data level by a straightforward join operation based on a common attribute or key, combining time-series data is not that simple.

In this paper, we present a novel approach to exploit data obtained from repeated GRN time-series measures under varying conditions to infer robust GRN models. We demonstrate the usefulness of this approach by comparing the resultant multi-condition GRN models with the individual single-condition models. At present it is still relatively costly and time consuming to perform multi-condition experiments, hence we base our study on artificial GRN time-series data sets. While this is a limitation in the present study, we believe that the results we obtained are still valid when applied to data derived from real GRN systems. It is likely that in the future the costs for multi-condition experiments will be lower and that multi-condition experiments are therefore expected to become commonplace.

II. Reverse engineering and data combination

Many reverse-engineering methods have been proposed in recent years. Furthermore, different approaches have been adopted in an attempt to improve the performance of these methods and create more robust models, improve predictive performance, and identify regulatory interactions from gene expression measurements. Swain et al. [5] evaluated three commonly used approaches which formulate dynamic GRN models as ordinary differential equations (ODEs). The authors assessed the ability of the different ODE structures to

978-1-4673-5875-0/13/$31.00

replicate the GRN systems' regulatory structure and dynamic gene expression behavior under varying conditions. The study evaluated three commonly encountered ODE rate law formulations: SS, ANN and GRLOT. The results suggest that the ANN and GRLOT methods are superior to the SS method in their ability to accurately predict network behavior. The former two methods also produced more accurate network structures from the underlying data than the SS method. However, the study did not explore combining data from multi-condition experiments. Although the research has thoroughly investigated the rate law formalisms, relying on sparse data based on only a single experiment may limit a model's usefulness in terms of the range of the experimental conditions that can be successfully predicted by it.

Andrews et al. [11] use an artificial intelligence model known as Cost Based Abduction (CBA) to generate GRN models from multiple data sources. They successfully apply their method to study the pheromone pathway in yeast using protein-DNA data, protein-protein interaction data and gene knock-out data. Their approach combines different data types which represent the same experimental condition. Yeang et al. [12] also explore the pheromone pathway in yeast by creating annotated interaction graphs they call *physical network models*. They combine multiple data types also. In our approach we consider only one gene expression data type, measurement of mRNA abundance, each data set representing the systems response to a different experimental condition.

Ting and Low [13] compare two approaches, *model combination* and *data combination* when multiple batches of data are available. The former method uses the available data sets creating a single model for each then combining the output. The latter approach creates one model, using the available data sets to train the model. The model combination method involves estimating predictive accuracy for a given instance through *k*-fold cross validation and creating a single model from the instances achieving the highest predictive performance (low error or high accuracy). They conclude that the model combination approach is stronger when there is only a small difference in the predictive error rate for each model. Our current study focuses on data combination utilizing all available data rather than a voting algorithm for the creation of a combined model.

Peeling and Tucker [14] present a method for modeling GRN's by forming a consensus Bayesian network model from multiple microarray gene expression data sets. Their study focuses on qualitative combination of Bayesian networks to determine the dependency structure between genes. The method of data combination presented in our study primarily focuses on reproducing quantitative network dynamics and predicting behavior. We have identified and discussed the need to develop this method further to include structure recovery in this article's section on future work.

Steele et al. [15] developed a method for transforming literature-based gene association scores to network prior probabilities. The Bayesian networks developed from this methodology therefore benefit from partial a priori knowledge of regulatory interactions thus simplifying the reverse-engineering process. Our current study reverse engineers GRN models using known network topology. The literature-based method therefore does not benefit the approach presented in this study but can be considered in future work concerning structure recovery.

Wang et al. [16] address the idea of combining multiple time-series microarray data sets by developing a high-level framework they refer to as the Gene Network Reconstruction tool. The idea is to combine solutions from each data set into an overall, consistent solution. The combination therefore occurs at the model level rather than the data level and the focus of their study is primarily on network structure.

Chen et al. [17] propose a two-step method for inferring GRN models from multiple data sets. First, they infer (optimize) a GRN structure (network topology) from each data set, which is then combined into a single, final network. They propose two methods based on computing the statistical mean and mode for each resulting topology. The second step consists of an additional optimization process that estimates the parameters of the combined GRN model after discovering its structure.

Gupta et al. [18] used multi-objective optimization to integrate different methods for reverse-engineering. To illustrate this, they used a combination of linear ODE and correlation-based methods, using data from time-course and gene inactivation (knock-out) experiments. The novel aspect in this approach is the combination of different inference methods into the same procedure, along with using heterogeneous sources of input data.

Marbach et al. [19] investigated the way in which ensemble networks resulting from reverse-engineering experiments could be used to "vote" the topology of a combined GRN model. For creating the ensemble network, they used an evolutionary method called analog genetic encoding. They ran 50 iterations of this procedure, each time retaining the network with the best fitness as part of the ensemble. Then, they used ensemble voting to generate a new network, showing that this network outperforms all initial members of the ensemble.

Our novel approach is to combine the data of the multi-condition experiments and reverse engineer a single model from this. Initially our method is concerned with reproducing network dynamics, structure recovery is subject to future research. As we use multiple data sets for training the model, we hypothesize that the resulting model should be more robust, more accurate and open to a wider scope of perturbation and behavior prediction than models generated from a single data set.

III. Approach and Study Design

The approach in this investigation assumes that the biological GRN system, *S*, under study is provoked with a set of n distinct stimuli or *conditions* $C_1 \neq C_2 \neq \ldots \neq C_n$ to elicit the corresponding dynamic gene-regulatory *responses* $\mathbf{R}_1, \mathbf{R}_2, \ldots, \mathbf{R}_n$. In this study, a condition C_i is defined as a set of data values specifying the initial perturbation of the system variables, representing gene expression quantities.

$$\mathbf{R}_i = \begin{bmatrix} r_{i11} & \cdots & r_{ij1} & \cdots & r_{iu1} \\ \vdots & & \vdots & & \vdots \\ r_{i1k} & \cdots & r_{ijk} & \cdots & r_{iuk} \\ \vdots & & \vdots & & \vdots \\ r_{i1m} & \cdots & r_{ijm} & \cdots & r_{ium} \end{bmatrix}$$

Each response $\mathbf{R}_i$ represents a set of u time-series of transcript concentrations/abundances, where u denotes the number of genes in S, and m the number of time points sampled for each gene, and r_{ijk} denotes a single measurement of gene j at time point k.

For simplicity, we assume the same number of measurements m for each gene in the considered time interval. In general, this is not necessary, though. Conceptually, we distinguish two types of scenarios: a concrete and an abstract GRN system.

Concrete GRN system: The reverse-engineered GRN model represents a *single concrete* GRN *system* derived from a single individual, e.g. the cell cycle gene regulation network of a particular mouse cell. In this case, the multiple conditions are applied at time points $T_1 < T_2 < \ldots < T_n$, and the time interval $\Delta T = T_{i+1} - T_i$ between consecutive conditions is chosen large enough for the system to fully "recover" from the provocation with condition C_i. This implies that the condition has no lasting effect on the system (e.g. does not destroy the system).

Abstract GRN system: The reverse-engineered GRN model represents a *single abstract* GRN *system* derived from a collection of individuals that are assumed to be similar in some important aspects. For example, the cell cycle gene regulation networks of eight mouse cells (from one or more mice). In this case, there is no restriction on the conditions applied and all conditions may be applied in parallel.

Once the response data has been obtained, the process is identical for reverse-engineering GRN models representing concrete and abstract GRN systems. Based on the n response data sets from n experiments (each applying a different stimulus), we

1. Randomly determine n_L training or learning data sets L and n_V validation data sets V, such that $n = n_L + n_V$ (typically: $n_L > n_V$):
   $L = L_1, L_2, \ldots, L_{n_L}$
   $V = V_1, V_2, \ldots, V_{n_V}$
2. Generate a combined training or learning data set Λ.
3. Reverse-engineer from each of the n_L training data sets a GRN model $M_1, M_2, \ldots, M_{n_L}$.
4. Reverse-engineer the final model M_Λ from the combined learning data set Λ.
5. Validate (determine average accuracy or error) the models generated in Steps (3) and (4) against the validation data sets V.

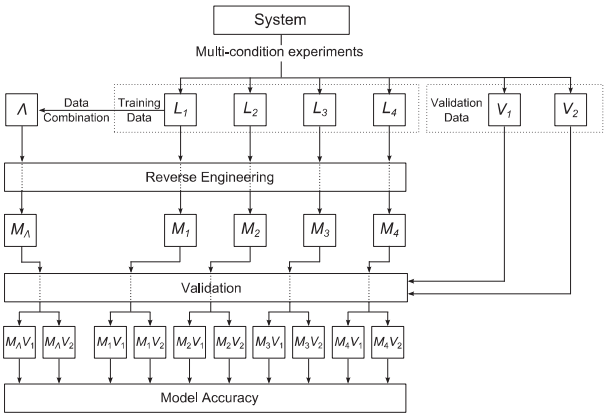

Fig. 1. Illustration of the process for reverse-engineering a GRN model from multi-condition experimental data (here with six conditions corresponding to four training or learning data sets and two validation data sets).

Fig. 1 depicts the basic study design we adopted to explore and evaluate our approach. In this particular case we used $n_L = 4$ and $n_V = 2$. In the diagram, the notation $M_i(V_j)$ denotes the data set created by simulating model M_i with the initial condition from data set V_j.

A. GRN Modeling

We use the term GRN systems to refer to gene-regulatory networks that describe regulatory gene-gene interactions without explicit representation of intermediary elements such as metabolites, nuclear receptors and transcription factors, which combine to direct and catalyze the reactions between genes [20]. The simulated data generated from the reference models represents measured mRNA abundance over time. Under this modeling assumption, one gene can either *activate* or *repress* another gene directly or indirectly (via other genes). Fig. 2 shows the network topology of the three GRN systems investigated in this study. System *A* describes a simple 3-gene [21], and *B* a 5-gene GRN system [22]. Model *C* describes a 7-gene GRN system based on the bile acid and xenobiotic system (BAXS) [23]. TABLE I. maps the nodes in the BAXS network to their corresponding genes and gene products.

TABLE I. ELEMENTS OF BAXS GRN NETWORK

Node	Gene	Gene product
X_1	*NR0B2*	SHP1
X_2	*NR1I2*	PXR
X_3	*NR1H4*	FXR
X_4	*ABCB1*	MDR1
X_5	*ABCC2*	MRP2
X_6	*ABCB11*	BSEP
X_7	*CYP3A4*	CYP3A4

The BAXS describes a genetic network that facilitates two distinct but intimately overlapping physiological processes; the enterohepatic circulation and maintenance of bile acid

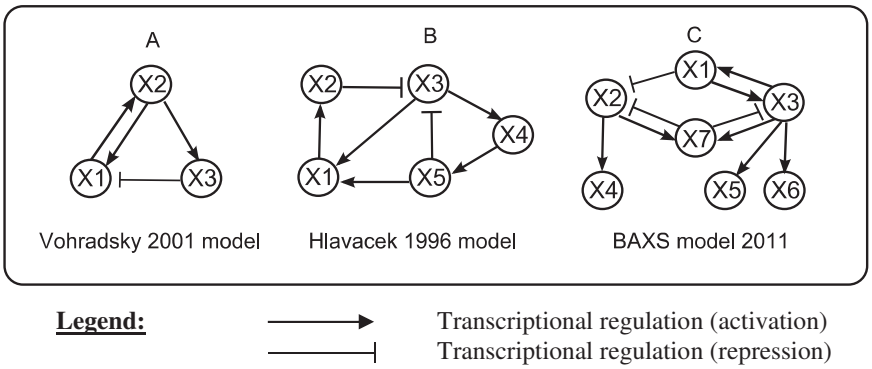

Fig. 2. The three gene regulatory networks or GRN system investigated in this study

concentrations and the detoxification and removal from the body of harmful xenobiotic, e.g. drugs, pesticides, and endobiotic compounds such as steroid hormones [24]. The model describes a simple catabolic pathway which is induced by the presence of a functional intermediate acting as an inducer of the network. Such an inducer could be either an endogenous or exogenous substance e.g. lithocholic acid (LCA), a secondary bile acid which activates transcription of both NR1I2 and NR1H4 [25] leading to activation of CYP3A4.This results in the production of enzymes which metabolize the inducer in this network e.g. LCA [26], thus switching off the network after a period of time. This is represented in the model as repression of NR1I2 and NR1H4 by CYP3A4 as the CYP3A4 enzyme metabolizes the inducer therefore there is no further activation of these genes.

B. Model equation and data generation

Common rate laws to model the reaction kinetics (regulatory interactions) of GRN systems include the *s-system*, *Hill functions*, *mass action kinetics*, *general rate law of transcription*, and *artificial neural network* formulations [4]. Because of its flexibility and advantageous properties [5], the models in this study are based on the ANN formalism [21]. Equation (1) defines the ANN-based rate of change X_i/dt of transcript X_i of gene i within a GRN system of u genes.

$$\frac{dX_i}{dt} = \frac{v_i}{1 + exp\left[-\left(\sum_{j=1}^{u} w_{ij}\, X_j + d_i\right)\right]} - k_i X_i \qquad (1)$$

where

- u defines the number of genes in the GRN system to be modeled, $i = 1, \ldots, u$.
- v_i denotes the maximal expression rate of gene i.
- X_j denotes the gene product of gene j influencing the product, X_i, of gene i, with: $j = 1, \ldots, u$.
- w_{ij} denotes the strength of control or regulation of gene j on gene i. Positive values indicate activating, negative values repressing control.
- d_i defines an *external* influence on gene i, which modulates the gene's sensitivity of response to activating or repressing influences. The higher $|\, d_i \,|$, the lower the influence of the weights w_{ij} on gene i. In GRN modeling, d_i is sometimes interpreted as *reaction delay parameter*, as it shifts the sigmoidal transfer function along the horizontal time axis, thus determining how fast the gene's expression level responds.
- k_i denotes the degradation rate constant of the i-th gene expression product.

Equation (1) defines a rate law capable of describing the dynamic behavior of GRN systems. The ANN rate law represents and calculates expression rates based on the weighted sum of multiple regulatory inputs. This additive input processing is able to represent logical disjunctions. The expression rate is restricted to a certain interval where the sigmoidal transfer function maps the regulatory input to the expression interval. The external input, d_i, regulates the sensitivity to the summed regulatory input of all genes.

For each of the three GRN systems depicted in Fig. 2, a single *GRN reference model* was manually created on the basis of the ANN rate law defined in Equation (1). Each GRN reference model serves as a surrogate for the corresponding biological GRN system to facilitate that generation of artificial dynamic gene expression data. The parameters for each reference model were determined manually through a process of trial and error. Each model was simulated with different parameter configurations multiple times using different initial conditions. After visual inspection of the generated data, the parameter values were updated and the reference models simulated again. This experimentation continued until plausible dynamics were identified. The criteria we applied to determine plausible dynamics were that a steady state was reached during the simulation, that no measurement increased infinitely nor all measured expression levels stabilized at zero.

We then used the three GRN reference models to create six time-series gene expression data sets based on different experimental conditions (initial values of transcript abundance). Visual inspection was again applied until six different system behaviors that were both plausible and sufficiently different were identified. The reference models were simulated over 600 sampling time points, each simulated Δt representing 1 second.

Through this process, we generated 6 time-series data sets for each of the three GRN systems representing the response of the systems to varying experimental conditions. From each of the three groups of six data sets, we randomly selected four as training or learning sets, and two as validation data sets. This configuration of data sets was viewed as a reasonable compromise: (a) on one hand, we required a number of training data sets under different conditions to be able to capture the underlying intricacies of the system under investigation, (b) on the other hand, we did not want to rely only on a single data set as independent validation data set.

Current time-series gene expression experiments typically sample somewhere between 10 and 30 time points, in rare cases several dozen time points are measured. After our initial exploration with 600 sampling points, we repeated the generation of 3 times 6 data sets, sampling 20 time points per gene over the explored total time interval. This sampling frequency provides more realistic data sets which are in line with current protocols for current time-series gene expression experiments.

By simulating the reference models over 600 sampling time points and then over 20 sampling time points, we can compare the accuracy of models reverse-engineered from both detailed and sparse data sets. This approach allows us to determine if the data combination method investigated in this work is sensitive to the amount of available data. Although these data sets have been created artificially, a great deal of thought went into this process, to ensure that the data is representative of current time-series gene expression experiments. We expect that the sampling frequencies and the number of conditions with which real GRN systems are being probed will continue to grow in the future.

C. Reverse engineering

Equipped with four training or learning data sets for each of the three GRN reference systems, we reverse engineered four *individual* GRN models for each system from the corresponding training data sets. Each reverse-engineering process is essentially a parameter estimation or optimization process applying the following algorithm.

Given a **learning data sets** L and the GRN model parameters $M = (\{ v_i \}, \{ w_{ij} \}, \{ d_i \}, \{ k_i \})$, where $i,j \in 1, \ldots, u$ (the number of genes in the model)

1. **Set and fix topology** (weight) parameters to zero for gene pairs that do not interact: $(\forall i,j \in \{1, \ldots u\}) (w_{ij} = 0 \mid no\_interaction(i,j))$.
2. **Initialize** remaining **parameters**. $M \leftarrow initialize$.
3. **Modify** non-fixed model **parameters** using particle swarm optimization: $M \leftarrow PSO$.
4. Use M to **simulate** time course data set, $\hat{Y}$, for all genes based on the initial values (condition) of L: $\hat{Y} \leftarrow simulate(L)$
5. **Compare** simulated data $\hat{Y}$ with the learning data L: $error \leftarrow compare(\hat{Y},L)$. IF *error* is not sufficiently small and maximum iterations are not reached, GO TO Step 3, otherwise finish learning and GO TO Step 6.
6. **Store** M as **GRN model**.

Notice, in Step 1 of the reverse-engineering algorithm described above, the ANN weights are fixed in such a way that for genes that are known *not* to interact, the weight is set to zero. These fixed weights are not subject to the optimization procedure (Step 3). The algorithm then proceeds to optimize or estimate the remaining ANN weights $\{ w_{ij} \neq 0 \}$ only for genes i and j that are known to interact in the underlying GRN system, plus the parameters $\{ d_i \}$ and $\{ k_i \}$. In other words, we assume that the interaction topology or network is known. This is a simplification to limit the computational complexity. In a future implementation of this algorithm, we will explore problems without this constraint.

For each learning data set, which represents a different condition on the same concrete GRN system, a GRN model was reverse-engineered. We used Copasi [27] to code and represent the GRN models and implement the reverse-engineer algorithm described above. To realize the parameter estimation step of the algorithm, we used Copasi's implementation of the particle swarm optimization (PSO) method [28]. Copasi also provides numerical integration methods needed in Step 4 of the algorithm, in which the time-course is predicted or simulated based on the initial values specified in the learning data set L. For deterministic solutions, the LSODA integrator is used [29]. The comparison Step 5 of the reverse engineering algorithm calculates the deviation (error) between each corresponding time-course in the learning data set L and the predicted or simulated time-course data $\hat{Y}$. To calculate the error, we applied the commonly used *root mean squared error* (*RMSE*) measure. Since our models consist of u genes, we essentially calculate the total RMSE by dividing the sum of all individual RMSEs by u.

The integration (simulation) Step 4 and the comparison Step 5 in the reverse-engineering algorithm is also relevant to the model validation stage, which comes after reverse-engineering has been completed.

D. Model validation

The training error is an indicator of how well the reverse-engineered model can replicate (simulate) the data from which it was constructed. A robust measure to assess how well the reverse-engineered model has captured the characteristics of the underlying system needs to determine the prediction error on unseen data. Because experiments generating gene expression time-course data are costly, the validation on independent data is frequently not reported in the literature. For each of the three GRN systems investigated in this study, have we use two independent validation data sets (V_1, V_2) to estimate the generalization error, and hence the

TABLE II. EXAMPLE OF COMBINING THE 4 DATA SETS OF THE VOHRADSKY 3 GENE MODEL. ONLY THE FIRST 6 TIMES STEPS ARE SHOWN.

	L_1			L_2			L_3			L_4		
Time	X_1^1	X_2^1	X_3^1	X_1^2	X_2^2	X_3^2	X_1^3	X_2^3	X_3^3	X_1^4	X_2^4	X_3^4
0	2.000	2.000	2.000	6.000	6.000	2.000	6.000	2.000	6.000	1.000	6.000	1.000
1	2.036	2.007	2.018	6.018	5.918	2.021	5.949	2.011	5.953	1.061	5.907	1.037
2	2.072	2.014	2.035	6.037	5.837	2.041	5.898	2.022	5.907	1.121	5.815	1.073
3	2.107	2.021	2.052	6.055	5.758	2.061	5.848	2.032	5.861	1.181	5.727	1.109
4	2.141	2.028	2.069	6.073	5.681	2.080	5.798	2.043	5.816	1.241	5.642	1.144
5	2.175	2.036	2.085	6.091	5.606	2.100	5.749	2.053	5.773	1.300	5.559	1.178
…	…	…	…	…	…	…	…	…	…	…	…	…

robustness, of our models, by averaging the total RMSE of the models on the two validation data sets.

E. Combined modeling algorithm

The steps described above explain the procedures we adopted to reverse-engineer and validate 3 times 4 individual models, 1 model for each of the 4 learning data sets for each of the 3 GRN systems depicted in Fig. 2. The training and validation errors we obtained are shown in TABLE III. The purpose of this study is to demonstrate that by using multi-condition experimental data, it is possible to generate more robust GRN models. Exploiting the multiple experimental data sets requires a suitable way of combining and using the available data to reverse-engineer a single model. The main steps of this algorithm are described below. We illustrate this based on the 3-gene Vohradsky GRN system.

1. Combining the learning data sets

For each of the $n_L = 4$ learning data set (L_1, …, L_4), the same number of gene expression abundances ($u = 3$) are sampled m times ($m = 20$) over the same total time interval. This means we can join these data sets as described in TABLE II, which shows the 4 data sets for the 3-gene Vohradsky GRN system were X_i^L refers gene i of data set L for the first 6 sampling points.

2. Formulation of combined model

Essentially, the first step of this combined-data GRN modeling algorithm creates a combined learning data set Λ with $n_L \times u$ genes or gene expression time-series. In the case of the 3-gene Vohradsky GRN system, we have a total number of $u_\Lambda = n_L \times u = 4 \times 3 = 12$ gene expression time-series (representing only 3 genes). Hence, we formulate a *combined-data* GRN model M'_Λ consisting of $u_\Lambda = 12$ variables representing 12 time-series but only 3 genes. (Notice, M'_Λ is not yet our final combined GRN model M_Λ) However, the combined model does not allow gene influences across the boundaries of the individual data sets; genes can only influence each other within the same individual training sets. Hence, using the ANN formulation in Equation (1), we get the following model specification for the 3-gene Vohradsky GRN system:

$$\frac{dX_1^1}{dt} = \frac{v_1}{1+exp\left[-\left(\sum_{j=1}^{u} w_{1j}\, X_1^1 + d_1\right)\right]} - k_1 X_1^1$$

$$\frac{dX_2^1}{dt} = \frac{v_2}{1+exp\left[-\left(\sum_{j=1}^{u} w_{2j}\, X_2^1 + d_1\right)\right]} - k_2 X_2^1$$

$$\frac{dX_3^1}{dt} = \frac{v_3}{1+exp\left[-\left(\sum_{j=1}^{u} w_{3j}\, X_3^1 + d_3\right)\right]} - k_3 X_3^1$$

$$\frac{dX_1^2}{dt} = \frac{v_1}{1+exp\left[-\left(\sum_{j=1}^{u} w_{1j}\, X_1^2 + d_1\right)\right]} - k_1 X_1^2$$

$$\frac{dX_2^2}{dt} = \frac{v_2}{1+exp\left[-\left(\sum_{j=1}^{u} w_{2j}\, X_2^2 + d_1\right)\right]} - k_2 X_2^2$$

$$\frac{dX_3^2}{dt} = \frac{v_3}{1+exp\left[-\left(\sum_{j=1}^{u} w_{3j}\, X_3^2 + d_3\right)\right]} - k_3 X_3^2$$

$$\frac{dX_1^3}{dt} = \frac{v_1}{1+exp\left[-\left(\sum_{j=1}^{u} w_{1j}\, X_1^3 + d_1\right)\right]} - k_1 X_1^3$$

$$\frac{dX_2^3}{dt} = \frac{v_2}{1+exp\left[-\left(\sum_{j=1}^{u} w_{2j}\, X_2^3 + d_2\right)\right]} - k_2 X_2^3$$

$$\frac{dX_3^3}{dt} = \frac{v_3}{1+exp\left[-\left(\sum_{j=1}^{u} w_{3j}\, X_3^3 + d_3\right)\right]} - k_3 X_3^3$$

$$\frac{dX_1^4}{dt} = \frac{v_1}{1+exp\left[-\left(\sum_{j=1}^{u} w_{1j}\, X_1^4 + d_1\right)\right]} - k_1 X_1^4$$

$$\frac{dX_2^4}{dt} = \frac{v_2}{1+exp\left[-\left(\sum_{j=1}^{u} w_{2j}\, X_2^4 + d_1\right)\right]} - k_2 X_2^4$$

$$\frac{dX_3^4}{dt} = \frac{v_3}{1+exp\left[-\left(\sum_{j=1}^{u} w_{3j}\, X_3^4 + d_3\right)\right]} - k_3 X_3^4$$

Notice, in the 12 equations describing the combined-data model M'_Λ, each set of model parameters appears only $u = 3$ times, i.e., once for each gene in the underlying GRN system (this is also the reason why u is constant across all 12 equations). Since the ANN model formulation (Equation (1)) defines $p = 6$ parameters (v_i, w_{i1}, w_{i2}, w_{i3}, d_i, k_i,) for each gene i and, in this case, we have 3 genes ($u = 3$), we get a total of $p \times u = 6 \times 3 = 18$ parameters to estimate/optimize in the reverse-engineering process (see next step). This

means that the number of parameters to be optimized for the combined-data model M'_A is independent of the number of conditions which create the multi-condition data sets.

3. Reverse-engineering of combined-data model

With the formulation of the combined-data model M'_A in place, the reverse-engineering algorithm described above is applied to determine/estimate the model's parameters. The advantage is that now we have more data to estimate the model parameters and hence are likely to produce more reliable estimates.

4. Creation of final combined model

The combined-data model M'_A specifies n_L times the number of model equations than the final combined GRN model M_A. However, the parameters are exactly the same in both models. Hence, once the parameters of M'_A are determined (see previous step), we simply use these parameters and transfer them to M_A to obtain our final GRN model.

IV. Results

For each GRN system, the initial conditions for the first experiment were set equal to emulate a basal transcription level. Subsequent experiments altered the initial conditions imitating the effect of increased transcription of one or two genes in the network. The parameters for each reference model were chosen manually to reproduce plausible dynamics for each case. An individual GRN model was reverse engineered for each of the 4 experimental conditions on the system used to create the learning data and a model resulting from the combined data approach was reverse engineered (see TABLE I). Each model was simulated over 600 sampling time points and again over 20 sampling time points per gene.

TABLE III lists the RMSE for all 30 models derived from the 3 artificial GRN systems (labeled A, B, C in the table). The RMSE is shown for each reverse-engineered model against both the learning and validation data sets for 600 and 20 time points, respectively. The *learning* RMSE (columns labeled RE) is less than 0.10 for 25 out of 30 models. Also noticed, for the GRN systems A and B the learning RMSE for 600 time points tends to be lower than that for 20 time points. The final column shows the average RMSE on the validation data sets for each model. Only 8 of the 30 reverse-engineered models managed to score an average validation RMSE below 0.10.

The main result is that the final combined model M_A consistently scored considerably lower average validation RMSEs than any of the individual models ($M_{1, ..., }M_4$) across all three reference GRN systems A, B and C (Fig. 2). This suggests that multi-condition experimental data (as described in this article) can be successfully used to produce accurate GRN models and as we have validated against two unseen data sets this indicates a measure of confidence in the models robustness.

V. Conclusion and Future Work

For each of the three GRN systems (Fig. 2), the reverse-engineered GRN models created using the data combination approach were significantly more accurate than any of the other models. The observed results indicate the data combination approach to reverse engineering results in more robust GRN models than reverse engineering models from single-condition data sets. This result is consistent across the two time resolutions (600 and 20 sampling points). From this observation we are confident that the data combination method is not dependent on detailed data and can be applied successfully to sparse data sets without any loss of performance. The study was conducted using 3, 5 and 7 gene GRN systems which were chosen as they exhibited the characteristics of clusters of co-expressed genes. Also, to simplify the reverse-engineering process, each was reverse-engineered using the known topology of the network. Future studies will investigate the use of this data combination method without known network topology and also on larger GRN systems of approximately 20-25 genes. In both cases we expect a considerable increase of required compute power due to the increase number of genes and parameters required. Ultimately this approach needs to be applied to real

TABLE III. Mean RMSE scores for each reverse-engineered models (individual and final combined). RE stands for "reverse-engineering" and indicates the RMSE on the learning data, V_1 and V_2 the RMSE on the validation data, and the average validation RMSE $(V_1 + V_2)/2$.

		600 data points				20 data points			
GRN	**Model**	L_i, M_A	V_1	V_2	$(V_1+V_2)/2$	**RE**	V_1	V_2	$(V_1+V_2)/2$
A	M_1	0.022	0.292	0.868	0.580	0.038	0.518	0.263	0.391
	M_2	0.054	0.510	0.590	0.550	0.036	0.318	0.257	0.288
	M_3	0.027	0.151	0.378	0.265	0.092	0.159	0.463	0.311
	M_4	0.084	0.351	0.480	0.416	0.189	0.385	0.225	0.305
	M_A	0.006	0.054	0.055	0.055	0.053	0.139	0.161	0.150
B	M_1	0.111	0.411	0.455	0.433	0.109	0.401	0.444	0.423
	M_2	0.002	0.029	0.029	0.029	0.145	0.653	0.633	0.643
	M_3	0.037	0.160	0.472	0.316	0.018	0.130	0.239	0.185
	M_4	0.009	0.268	0.187	0.228	0.011	0.804	0.701	0.753
	M_A	2.88e-6	3.60e-4	3.60e-4	3.60e-4	1.215e-5	1.74e-4	1.73e-4	1.73e-4
C	M_1	0.019	0.059	0.071	0.065	0.022	0.128	0.090	0.109
	M_2	0.030	0.166	0.122	0.144	0.146	0.148	0.224	0.186
	M_3	0.034	0.302	0.076	0.189	0.028	0.271	0.075	0.173
	M_4	0.093	0.083	0.189	0.136	0.018	0.017	0.085	0.051
	M_A	0.006	0.019	0.006	0.013	0.006	0.018	0.011	0.015

biological data to assess how sensitive it would be to the noise expected in real biological conditions. However simulated data was used due to the lack of suitable data available. As such data becomes available in the future this can be done.

Swain et al. [5] suggest a two stage approach for recovering network structure from gene expression measurements. The next study will investigate combining this approach with the data combination method to assess if multiple data sets can improve the recovery of network structure. A follow-up study will then combine the methods for network recovery and behavior prediction and evaluate this as an overall approach to utilizing multiple data sets in modeling from gene expression measurements. Throughout this study the ANN formalism has been used to calculate expression rates. Other general rate laws such as the Hill equation [30], GRLOT [6], SS [31] and general mass action will be assessed with the data combination method to determine which is more accurate at inferring network structure, predicting dynamic behavior or both.

ACKNOWLEDGEMENTS

This work received funding from the EC's Seventh Framework Program (FP7/2007-2013) under grant agreement n° RI-261507 and also from the Department for Employment and Learning, Northern Ireland.

REFERENCES

[1] E. Davidson and M. Levin, "Gene regulatory networks," *Proceedings of the National Academy of Sciences of the United States of America,* vol. 102, no. 14, pp. 4935-4935, April 05 2005.

[2] M. Ronen, R. Rosenberg, B. I. Shraiman and U. Alon, "Assigning numbers to the arrows: Parameterizing a gene regulation network by using accurate expression kinetics," *Proceedings of the National Academy of Sciences,* vol. 99, no. 16, pp. 10555-10560, August 06 2002.

[3] Z. Bar-Joseph, "Analyzing time series gene expression data," *Bioinformatics,* vol. 20, no. 16, pp. 2493-2503, November 01 2004.

[4] K.-H. Cho, S.-M. Choo, S. H. Jung, J.-R. Kim, H.-S. Choi and J. Kim, "Reverse engineering of gene regulatory networks," *Systems Biology, IET,* vol. 1, no. 3, pp. 149-163, 2007.

[5] M. Swain, J. Mandel and W. Dubitzky, "Comparative study of three commonly used continuous deterministic methods for modeling gene regulation networks," *BMC Bioinformatics,* vol. 11, no. 1, p. 459, 2010.

[6] P. Mendes, W. Sha and K. Ye, "Artificial gene networks for objective comparison of analysis algorithms," *Bioinformatics,* vol. 19, no. suppl 2, p. 122, September 27 2003.

[7] M. Stetter, B. Schurmann and M. Djori, "Systems Level Modeling of Gene Regulatory Networks," in *Artificial Intelligence Methods and Tools for Systems Biology*, W. Dubitzky and F. Azuaje, Eds., Springer, 2004, pp. 175-195.

[8] C. Spieth, F. Streichert, N. Speer and A. Zell, "Inferring Regulatory Systems with Noisy Pathway Information,," in *German Conference on Bioinformatics (GCB 2005)*, Hamburg, Germany, 2005.

[9] A. Fuente and D. P. Makhecha, "Unravelling gene networks from noisy under-determined experimental perturbation data," *Systems Biology, IEE Proceedings,* vol. 153, no. 4, pp. 257-262, 2006.

[10] D. L. Hall and J. Llinas, "An introduction to multisensor data fusion," *Proceedings of the IEEE,* vol. 85, no. 1, pp. 6-23, 1997.

[11] E. A. M. Andrews and A. J. Bonner, "Explaining Genetic Knock-Out Effects Using Cost-Based Abduction," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, Barcelona, 2011.

[12] C. Yeang, T. Ideker and T. Jaakkola, "Physical Network Models," *Journal of Computational Biology,* Vols. 11(2-3), pp. 243-262, 2004.

[13] K. Ting and B. Low, "Model combination in the multiple-data-batches scenario," *Lecture Notes in Computer Science,* vol. 1224/1997, pp. 250-265, 1997.

[14] E. Peeling and A. Tucker, "Consensus gene regulatory networks: combining multiple microarray gene expression data sets," *AIP Conference Proceedings,* vol. 940, no. 1, pp. 38-49, September 18, 2007 2007.

[15] E. Steele, A. Tucker, P. Hoen and M. J. Schuemie, "Literature-based priors for gene regulatory networks," *Bioinformatics,* vol. 25, no. 14, pp. 1768-1774, July 15 2009.

[16] Y. Wang, T. Joshi, X.-S. Zhang, D. Xu and L. Chen, "Inferring gene regulatory networks from multiple microarray data sets," *Bioinformatics,* vol. 22, no. 19, pp. 2413-2420, October 01 2006.

[17] B.-L. Chen, L.-Z. Liu and F.-X. Wu, "Inferring gene regulatory networks from multiple time course gene expression data sets," in *2011 IEEE International Conference onSystems Biology (ISB)*, Zhuhai, China, 2011.

[18] R. Gupta, A. Stincone, P. Antczak, S. Durant, R. Bicknell, A. Bikfalvi and F. Falciani, "A computational framework for gene regulatory network inference that combines multiple methods and data sets," *BMC Systems Biology,* vol. 5:52, 2011.

[19] D. Marbach, C. Mattiussi and D. Floreano, "Combining Multiple Results of a Reverse-Engineering Algorithm: Application to the DREAM Five-Gene Network Challenge," *Annals of the New York Academy of Sciences,* vol. 1158, no. 1, pp. 102-113, 2009.

[20] G. Karlebach and R. Shamir, "Modelling and analysis of gene regulatory networks," *Nature reviews. Molecular cell biology,* vol. 9, no. 10, pp. 770-780, 10//print 2008.

[21] J. Vohradsky, "Neural network model of gene expression," *The FASEB Journal,* vol. 15, no. 3, pp. 846-854, March 01 2001.

[22] W. S. Hlavacek and M. A. Savageau, "Rules for Coupled Expression of Regulator and Effector Genes in Inducible Circuits," *Journal of Molecular Biology,* vol. 255, no. 1, pp. 121-139, 1/12 1996.

[23] N. Kennedy, P. Thompson, H. Zheng and W. Dubitzky, "Multi-Scale modelling of the Bile Acid and Xenobiotic System," in *Proceedings of The 2011 International Conference on Bioinformatics & Computational Biology*, USA, 2011.

[24] S. A. Kliewer and T. M. Willson, "Regulation of xenobiotic and bile acid metabolism by the nuclear pregnane X receptor," *Journal of lipid research,* vol. 43, no. 3, pp. 359-364, March 1 2002.

[25] D. D. Moore, S. Kato, W. Xie, D. J. Mangelsdorf, D. R. Schmidt, R. Xiao and S. A. Kliewer, "International Union of Pharmacology. LXII. The NR1H and NR1I Receptors: Constitutive Androstane Receptor, Pregnene X Receptor, Farnesoid X Receptor α, Farnesoid X Receptorβ, Liver X Receptor α, Liver X Receptor β, and Vitamin D Receptor," *Pharmacological Reviews,* vol. 58, no. 4, pp. 742-759, December 2006.

[26] S. Nishida, J. Ozeki and M. Makishima, "Modulation of Bile Acid Metabolism by 1α-Hydroxyvitamin D_3 Administration in Mice," *Drug Metabolism and Distribution,* vol. 37, no. 10, pp. 2037-2044, October 2009 October 2009.

[27] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes and U. Kummer, "COPASI–a COmplex PAthway SImulator," *Bioinformatics,* vol. 22, no. 24, pp. 3067-3074, December 15 2006.

[28] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Micro Machine and Human Science, 1995. MHS '95., Proceedings of the Sixth International Symposium on*, 1995.

[29] L. Petzold, "Automatic Selection of Methods for Solving Stiff and Nonstiff Systems of Ordinary Differential Equations," *SIAM Journal on Scientific and Statistical Computing,* vol. 4, no. 1, pp. 136-148, 1983.

[30] H. D. Kim and E. K. O'Shea, "A quantitative model of transcription factor-activated gene expression," *Nat Struct Mol Biol,* vol. 15, no. 11, pp. 1192-1198, 2008.

[31] M. A. Savageau, "Biochemical systems analysis. A study of function and design in molecular biology. Addison-Wesley, Reading Mass. 1976.," *Biochemical education,* vol. 5, no. 4, pp. 84-84, 1977.