A multi-model reverse-engineering algorithm for large gene regulation networks

Alexandru Mizeranschi, Paul Thompson, Huiru Zheng, Werner Dubitzky
University of Ulster, Coleraine/Jordanstown, UK

***Abstract*—Modeling and simulation of gene-regulatory networks (GRNs) has become an important aspect of modern systems biology investigations into mechanisms underlying gene regulation. An important and unsolved problem in this area is the automated inference (reverse-engineering) of dynamic, mechanistic GRN models from time-course gene expression data. The conventional *one-stage* model inference algorithm determines the values of all model parameters simultaneously. Recently, *two-stage* algorithms have been proposed to improve the accuracy of the inferred models and the efficiency of the reverse-engineering process. The main objective of this study is to compare the performance of the conventional one-stage and the modern two-stage algorithm, with emphasis on the computational complexity. We explored data generated from artificial and real GRN systems under different experimental conditions and regulatory structure constraints. Our results suggest that the 2-stage approach outperforms the one-stage methods by far in terms of model inference speed without a loss of accuracy.**

I. Introduction

The regulation of genes and their products is at the heart of a systems view of complex biological processes. Hence, the modeling and simulation of *gene-regulation networks* (*GRNs*) is becoming an area of growing interest in systems biology research [1]. As the amount of gene expression data is growing, researchers are becoming increasingly interested in the *automated inference* or *reverse-engineering* of quantitative dynamic, mechanistic gene-regulatory network *models* from time-course gene expression data [2]. The quality of such reverse-engineered GRN models is determined mainly by two factors: (1) **Predictive power**: The accuracy of *predicted time-course responses* for unseen stimulus/input data (i.e. new experimental/biological conditions). (2) **Inferential power**: The accuracy of the reverse-engineered gene-regulatory *structure*.

Reverse-engineering GRN models with high predictive and inferential power is a long-standing problem [2]. In the present study, we focus on mechanistic models based on *ordinary differential equations* (*ODEs*) that represent both the gene expression kinetics of the system and its regulatory structure features. Such approaches represent a GRN system consisting of N genes by N ODEs. Common equation forms, or rate laws, include the S-system (Eq. (2)) and the ANN (Eq. (6)).

The maximum number of regulatory interactions in a GRN system is N^2, where N denotes the number of genes in the system. The number of model parameters to be determined in the inference process grows as the number of genes N increases. The number of model parameters also depends on the rate law chosen to represent the system. For models with a large number of free parameters the computation effort/time needed to determine the model parameters becomes an important factor. Real gene networks are sparse, with an average number of ca. 2 regulators per gene [3]. Viewing *sparseness* of GRNs as a "design constraint", GRN inference approaches could use this to improve the effectiveness and efficiency of the reverse-engineering process.

Two basic model inference algorithms have been proposed in the literature: the conventional *1-stage algorithm* and a more recent *2-stage algorithm* [4], [5], [6], [7]. 1-stage algorithms attempt to estimate all model parameters simultaneously. By contrast, 2-stage approaches first construct an ensemble of N one-gene GRN models (one model per gene), derive the overall gene-regulatory structure from the ensemble, and then estimate the remaining model parameters in a final inference process.

The 2-stage approach has some important advantages over the 1-stage algorithm. Essentially, the 2-stage procedure realizes a kind of ensemble strategy. The advantage of such a strategy is that the intermediate information (generated in stage 1) can potentially be exploited to improve the overall model construction process. For instance, this could be used to improve the all-important inferential power, which is the major weakness of state-of-the-art inference algorithms. In this study, we do not seek to improve inferential or predictive power over state-of-the-art methods. Instead, this study focuses on assessing the computational complexity of the two methods.

II. Related work

A. 2-stage GRN model inference approaches

The 2-stage approach was first proposed by Maki et al. [7]. It decomposes the system of ODEs into N individual equations and infers the parameters corresponding to one gene at a time. In this way, the number of parameters per single-gene model is reduced by a factor of N, where N is the number of genes in the GRN system. The $i = 1, ..., N$ resulting single-gene models, Eq. (1), have a form similar to Eqs. (2) and (6).

$$\frac{dy_i(t)}{dt} = f_i(y_1(t), y_2(t), ..., y_N(t)) \tag{1}$$

where

$$y_i(t) = \begin{cases} x_j(t) & \text{if } j = i \\ \hat{x}_j(t) & \text{otherwise} \end{cases}$$

and $\hat{x}_j$ is obtained by making a *direct estimation* on the *observed* gene expression level of gene j from the available time-series data using a technique such as spline interpolation. This direct estimation procedure becomes necessary because the numerical integration procedure (ODE solver) requires estimates of the the input data for all inputs at a temporal resolution which is usually much higher than that of the experimental data. When we reverse-engineer the full model with the 1-stage procedure, these estimates are provided by the solutions of the other ($j \neq i$) equations in the ODE system. A positive side effect of this direct estimation procedure is that the degree of model-data fit of single-gene models is very high.

Swain et al. [4] performed a comparative study of three commonly used ODE rate laws: the ANN rate law [8], the S-system [9], and the generalised rate law of transcription [10]. They employed a 2-stage approach to infer GRN models and compared these in terms of inferential and predictive power. A genetic algorithm was used as optimization method. In the first stage (called *bottom-up phase*), the optimization problem is split into N gene-specific parameter inference problems that infer the parameters of single-gene models. The algorithm retains the fittest single-gene models after a number of iterations. These models are then combined into a full GRN model. The GRN models they created with this approach (from single stimulus/response data sets) were able to reproduce the training data accurately. However, an accurate reproduction of regulatory network features was only possible with training data originating from multiple experiments under varying conditions.

B. Penalty term approaches

Kikuchi et al. [11] proposed to infer biologically plausible (sparse) GRN structures by incorporating prior knowledge into the GRN model inference procedure. Essentially, sparseness relates to the empirical observation that the total fraction of regulatory influences in biological GRN systems is far lower than the maximal number, N^2, of possible regulatory influences. The authors used the S-system rate law formulation (Eq. (2)):

$$\frac{dx_i(t)}{dt} = \alpha_i \prod_{j=1}^{N} x_j^{g_{ij}}(t) - \beta_i \prod_{j=1}^{N} x_j^{h_{ij}}(t), \text{ with } i = 1...N \quad (2)$$

where N denotes the number of genes in the GRN system, α_i and β_i denote constants for transcript synthesis and degradation rates, and g_{ij} and h_{ij} denote the kinetic orders governing the synthesis- and degradation-regulatory influence of gene j on gene i, such that positive values indicate activation, negative values indicate repression, and zero values indicate absence of any regulation.

Their approach defines the *error function* shown in Eq. (3) to enforce GRN structures with higher degrees of sparseness. The error function computes the combined deviation (error) of the predicted versus observed experimental time courses. The first term of the error function defines a relative squared error, the second is the penalty term.

$$\Delta = \sum_{k=1}^{N} \sum_{t=1}^{T} \left(\frac{\hat{x}_k(t) - x_k(t)}{x_k(t)} \right)^2 + C \left(\sum_{i,j} |g_{ij}| + \sum_{i,j,i \neq j} |h_{ij}| \right) , \quad (3)$$

where $i, j, k = 1...N$ identify the genes in the system, $x_k(t)$ denotes the experimentally observed and $\hat{x}_k(t)$ the predicted amount or concentration of transcript of gene k at time t; g_{ij} and h_{ij} denote the kinetic orders of the S-system rate law (Eq. (2)); and C denotes the penalty constant defining how much the penalty term contributes to the the overall error. A value of $C = 0$ means no penalty is added. The sums in the penalty term essentially add up the magnitude or strength of the regulatory influences (repressing and activating) from genes $j = 1, ..., N$ to gene i.

The penalty term of the error function in Eq. (3) effectively enforces a sparse connectivity edge structure in the inferred GRN model graph – fewer regulatory influences with lower regulatory strength result in a lower overall error computed by the Δ function. Kikuchi et al. have tested their approach for C values between 0.15 and 0 (no penalty). They found that the convergence rate of their experiments with the penalty term was 5 times higher than for experiments without penalty. This was also reflected in lower computation times: parameter inference experiments with $C = 0.15$ were 1.5 times faster than those with no penalty ($C = 0$).

Kimura et al. [5] combined the two-stage approach of Maki et al. (Eq. (1)) with the sparsity-inducing penalty term of Kikuchi et al. (Eq. (3)). They observed that the penalty term approach is biased as it has a tendency to "push" the number of regulatory influences to zero. They proposed an alternative formulation of the penalty term that compensates for this inherent bias. In their formulation, the penalty term contains a parameter specifying the "desired" average number of regulatory influences. Thus, the error function for single-gene models corresponding to gene i can be stated as follows:

$$\Delta_i = \sum_{t=1}^{T} \left(\frac{\hat{x}_i(t) - x_i(t)}{x_i(t)} \right)^2 + C \sum_{j=1}^{N-I} (|G_{ij}| + |H_{ij}|) , \quad (4)$$

where I denotes the average number of regulatory influences, $|G_{ij}|$ and $|H_{ij}|$ denote vectors that arrange the absolute values of the kinetic orders g_{ij} and h_{ij} in ascending order, and the rest of the parameters have the same interpretation as in Eq. (3). In this formulation, the penalty term only penalizes the error in cases where the number of genes which directly affect gene i (either by synthesis regulation or by degradation) is greater than I.

Noman and Iba [6] further improved the penalty term formulation of the gene-specific error function Δ_i in Eq. (4) by grouping the kinetic orders g_{ij} and h_{ij} into a single term $|K_{ij}|$:

$$\Delta_i = \sum_{t=1}^{T} \left(\frac{\hat{x}_i(t) - x_i(t)}{x_i(t)} \right)^2 + c \sum_{j=1}^{2N-I} (|K_{ij}|) , \quad (5)$$

where $|K_{ij}|$ denotes a vector of the combined kinetic orders g_{ij} and h_{ij} in ascending order. The error function in Eq. (5) combines the kinetic orders into a single vector (sorted in ascending order) to avoid a bias inherent in the error function in Eq. (4) towards regulations where a regulator gene affects the target gene both at the level of synthesis (g_{ij}) and degradation (h_{ij}). Because it is uncommon for a gene to regulate both the synthesis and degradation of its target gene, the error function of Noman and Iba seems more realistic. The authors also adapted this approach to the ANN rate law [12]. However, their results varied considerably from one experiment to the next. This was attributed to the under-determination of the problem (lack of sufficient amounts of data) and the noise inherent in the gene expression measurements.

Because of the promising results achieved with the penalty term approach and the ability of this method to capture the sparseness of the regulatory structure inherent in real GRN systems, we adopted this approach in our own experiments.

III. Approach

A. Methods

For the experiments presented in this study, we have adopted the ANN rate law proposed by Vohradský [8] to represent GRN models because this formalism has the advantage of having fewer parameters to optimize than comparable rate laws [8], [4], [9], [10]. The ANN rate law is defined by Eq. (6).

$$\frac{dx_i(t)}{dt} = \frac{a_i}{1 + exp(-\sum_j^N \omega_{ij} x_j(t) + \theta_i)} - d_i x_i(t), \quad (6)$$

where $i, j = 1, ..., N$ denote genes in a GRN system consisting of N genes, and $x_i(t), x_j(t)$ denote the total amount or concentration of transcript of genes i and j, respectively, at time t. Furthermore, a_i denotes the maximal rate of transcript synthesis of gene i, and d_i is a rate constant controlling the rate of transcript degradation. The parameter ω_{ij} represents the strength of the synthesis-regulatory influence of gene j on gene i, where $\omega_{ij} > 0$ represents synthesis activation, $\omega_{ij} < 0$ synthesis repression, and $\omega_{ij} = 0$ means no synthesis regulation. The parameter θ_i represents an activation/repression threshold for the sigmoidal regulation function regulating the messenger synthesis of gene i.

In order to facilitate the inference of realistic GRN models with sparse a network structure, we adopted the penalty approach defined by Eq. (5) [12]. We used the *normalised root mean squared error* (*NRMSE*) to determine the deviation of the time courses predicted by the GRN model and the experimentally observed data. The NRMSE is less sensitive to the absolute magnitude of the measurements, hence it facilitates comparison of errors across models and systems. The NRMSE is defined by the root mean squared error divided by the range of the observed values.

The quantitative effects that the GRN structure information has on the accuracy and speed of reverse-engineering GRN models has previously been assessed [13]. If no information about the GRN topology is used in the reverse-engineering process (i.e. proportion regulators, regulators per gene, etc.), the optimization process considers each gene as a possible regulator of all other genes in the GRN system. It was found that model inference is much faster and more accurate if the network topology of the reverse-engineered system is known. This study focused on quantitatively comparing the performance (computing time, model errors) of reverse-engineering GRN models with the "monolithic" 1-stage and the modular 2-stage approach. Because the ω_{ij} parameters of the ANN rate law defined by Eq. (6) are implemented as double precision numbers in our implementation, the chance that the numerical procedures will result in exactly $\omega_{ij} = 0$ is extremely low. Thus, we adopted a threshold approach reported by Tominaga et al. [14] as follows: if $|\omega_{ij}| < \alpha$, then we set $\omega_{ij} = 0$. We used a threshold of $\alpha = 0.1$ in our experiments.

In order to estimate the model parameters, we used the *particle swarm optimization* (*PSO*) algorithm [15]. The PSO is a population-based meta-heuristic inspired by the flocking, schooling or swarming behavior of animals. Two main advantages of this method include that it optimizes continuous variables and it has the ability to avoid getting stuck in local minima by using a multi-swarm approach which successively swaps particles across each swarm after a fixed number of iterations in order to increase the "genetic" diversity of the overall swarm.

B. GRN systems and data

We used 6 artificial and 1 real GRN system to explore and compare the performance characteristics of the 1-stage and 2-stage model inference process. With the JAGN tool [16] we manually created six artificial GRN reference systems, three 11-gene and three 22-gene systems, with an average regulatory connectivity of 2, 3 and 4, respectively, in each of the groups of three systems. We simulated each reference system under two initial conditions to produce two data sets (a training and corresponding validation data set). In order to create time series data that is as similar to real gene expression data, we added Gaussian noise with a standard deviation of 0.15 to create three noisy versions for each data set. This resulted in a total of 120 data sets, 60 used for model inference (training) and 60 for model validation. We reverse-engineered three GRN models from each of the 60 training data sets, varying the average connectivity parameter value I from 2 to 3 and then 4. This gave us 180 reverse-engineered models for each reverse-engineering technique, 360 in total.

In order to assess the behavior of the two modelling approaches on real biological data, we have set up a smaller number of experiments using data sets published by Pramila et al. [17]. We used the alpha 30 and alpha 38 data sets. As a reference model, we focused on a network with 11 genes and 34 regulatory interactions – the yeast cell cycle "core" network, initially described by Li et al. [18] and reverse-engineered by Böck [19]. We executed three sets of experiments, each containing three experimental repeats. The algorithmic average connectivity varied across the three experimental sets with the values 2, 3 and 4, using the 1-stage and 2-stage approaches. In total, we performed 18 experiments using this biological data set.

C. Experimental set-up

We carried out a set of reverse-engineering experiments designed to compare the 2-stage method to the "conventional" (monolithic) 1-stage reverse-engineering approach which infers complete GRN models without considering single-gene models in a first, separate step. In order to provide a fair comparison between the two methods, we have set the termination criteria as follows. The main PSO algorithm was set to iterate for a maximum of 200 particle exchanges across the PSO islands or swarms, with such an exchange occurring every 50 PSO iterations and swapping 4 particles from each island to the next one in a circular topology. The size of a sub-swarm was set to 50 particles and the total swarm consisted of 50 such sub-swarms. The PSO algorithm converges if the optimization error Δ drops below 0.15. These values were empirically chosen such that they would ensure the successful termination of each experiment.

For the first stage of the 2-stage approach, the PSO setting was altered as follows: each PSO swarm contained 20 particles, the total number of PSO iterations was set to 200, and the error threshold was 0.05. A similar configuration change was adopted for the real-data experiments, where we increased the sub-swarm size to 200 particles in order to facilitate the successful convergence of these experiments.

We used the following criteria to evaluate the performance of reverse-engineering experiments and the inferred models: **Wall time**: The total time to reverse-engineer a model. For the 2-stage experiments, this includes the computation times of both stages. **Number of optimization iterations**: The number of particle migrations performed by PSO during the second reverse-engineering stage. **Training error threshold**: Threshold used to terminate the model inference process. After some explorative experimentation, we used a training threshold of 0.15. **Validation error**: Each inferred model was validated against an independent validation data set, and the model validation error was recorded. **F-Score**: A value or score characterizing the *inferential power* of the inferred models, i.e. the accuracy of the inferred GRN regulatory structure. We adopted the F-score approach reported by Mazur et. al [20] for a three-class (activation, repression, no regulation) problem.

D. Implementation and hardware set-up

The 2-stage approach presented in this study was realized with the MultiGrain tool which was developed in-house and described elsewhere [13]. This tool implements a multi-island or multi-swarm version of the PSO algorithm [15] which allows for the possibility for parallelization by domain decomposition. This is achieved by creating multiple instances of a software module which implements a PSO swarm and synchronizing these via message passing using the MUSCLE [21] coupling library. In terms of the 2-stage algorithm, the implementation allows for multiple PSO threads to be executed during the first (decoupled ODE) stage. Using multiple CPU cores, MultiGrain is able to infer multiple GRN structures corresponding to the number of PSO sub-swarms and select the optimal topology (having the lowest training or validation error) to fix the structure parameters for the second reverse-engineering stage. The ODE solver in MultiGrain relies on Michael Thomas Flanagan's Java Scientific Library, which implements the Runge-Kutta 4 algorithm used for ODE integration and cubic spline interpolation method used for estimating the transient gene product concentrations in the single-gene model integration phase of the 2-stage approach.

MultiGrain has been tested before in a distributed environment [21] and it was found that the tool is able to efficiently make us of the increased computing power available by increasing the number of PSO sub-swarms and total swarm size. For the current study, the hardware set-up consisted of a Linux machine with 48 AMD Opteron 8431 processors, each with 6 cores. The total amount of RAM was 500 GB.

IV. Results and discussion

We notice that the 2-stage algorithm reverse-engineering experiments were indeed much faster than the ones employing the one-stage approach. For example, based on an average system connectivity of 3, the average (across enforced average reverse-engineering connectivities 2, 3 and 4) 1-stage reverse-engineering wall time was 84.07 seconds, and for the 2-stage version 19.43 seconds. Thus, the 2-stage algorithm was more than 4 times faster. This difference was even greater for the 22-gene case studies, where the computation times were 20 times lower (42.07 seconds for an average system connectivity of 3) for the 2-stage reverse-engineering jobs than for the corresponding 1-stage versions (868.33 seconds). The computation times improved also with an increase of the average connectivity enforced by the penalty term, signifying that the algorithms were more successful at finding suitable solutions when the imposed connectivity restriction was less demanding. This difference was most notable for the 11-gene experiments using the 1-stage algorithm, where scenarios having an enforced reverse-engineering connectivity of 4 were completed successfully in half of the time taken for the scenarios with connectivity 2: 57.7 seconds vs. 97.6 seconds on average. These differences were however much less notable for the 22-gene scenarios.

A proportional decrease of the number of PSO iterations was observed in the 2-stage approach when compared with the 1-stage approach. The number of iterations performed was roughly between 10 and 20 times lower for the 11-gene experiments, with a lower decrease for the scenarios involving artificial systems with average connectivity 2 (from 46.33 to 4.8 iterations on average for the experiments with average connectivity 2) and a higher difference for larger average connectivities (38.87 to 2.6 and 23.8 to 1 iteration for average connectivities 3 and 4, respectively). Similarly, the number of iterations dropped by 30 to 50 times for the 22-gene scenarios, from between 50 and 60 iterations to only one or two iterations on average. This signifies that the first stage of the 2-stage algorithm was able to find GRN structures that increased the performance of the second reverse-engineering stage in terms of the number of iterations required for convergence. The question remains, however, whether the resulting GRN models were meaningful in terms of their predictive and inferential power.

The training errors, validation errors and the F-score values were generally similar for the two reverse-engineering methods. The cut-off value of 0.15 for the training error was marginally exceeded during the 2-stage scenarios, which achieved training errors as low as 0.13 on average. In the case of validation errors, their values did not show a consistent trend depending on the choice of reverse-engineering algorithm and average connectivity. However, these values were lower when the 22-gene systems were being targeted.

With an increase in the average connectivity enforced by the reverse-engineering algorithm, we noticed a considerable variability in F-score values, which may cast doubt on the practical usefulness of this penalty term. For all values of artificial system average connectivity, the F-score increased on average 1.5-fold and 2-fold for the experiments with average connectivities of 3 and 4, respectively, compared to the experiments using the value 2. The F-score increase was likely caused by the larger degrees of freedom for the reverse-engineering algorithm for larger values of the penalty term.

Similar observations were made for the experiments with real biological data. We noticed a 3-fold decrease in computation times for the 2-stage (123.56 seconds) compared to the 1-stage experiments (43.67 seconds), while the iteration numbers decreased by almost 4 times, from 21.44 to 5.67 iterations on average. These differences are not as pronounced as for experiments with artificial data. This may be explained by the increase of the PSO swarm size aimed at improving the success rate at stage 2 of the 2-stage process, which also had the effect of reducing the overall time. The average training errors, validation errors and F-scores were similar, which paints a similar picture as for the artificial data experiments.

V. Conclusions

Automated inference of gene-regulation models from gene expression time-course data is a long-standing and unsolved problem [2]. Such models are useful, provided they perform well on unseen data (predictive power) and that their regulatory structure is close to the real structure of the underlying GRN system (inferential power). Developing methods with high inferential power is hampered by a lack of data on the regulatory structure of real biological systems. Furthermore, automated GRN model inference for *large* GRN systems constitutes a considerable computational challenge because of the large number of parameters to be estimated. This study focused on a comparison of the 2-stage and a 1-stage reverse-engineering approaches, based on the hypothesis that the former would outperform the latter in terms of computational efficiency, without a significant loss in predictive and inferential power.

We found that 2-stage was considerably faster than 1-stage for both artificial and real biological data. We quantified the difference in terms of wall time and the number of PSO iterations. No notable difference in terms of predictive and inferential power was observed. In future studies, we intend to explore approaches aiming to improve the inferential power of models reverse-engineered using MultiGrain. Approaches employing GRN model ensembles [22], [2] and empirical knowledge derived from real GRN systems seem to be promising lines of future research in this area.

References

[1] G. Karlebach and R. Shamir, "Modelling and analysis of gene regulatory networks," *Nat. Rev. Mol. Cell Biol.*, vol. 9, no. 10, pp. 770–780, 2008.

[2] D. Marbach, J. C. Costello, R. Küffner, N. M. Vega, R. J. Prill, D. M. Camacho, K. R. Allison, T. D. Consortium, M. Kellis, J. J. Collins, and G. Stolovitzky, "Wisdom of crowds for robust gene network inference," *Nature Methods*, vol. 9, pp. 796–804, 2012.

[3] R. D. Leclerc, "Survival of the sparsest: Robust gene networks are parsimonious," *Molecular Systems Biology*, vol. 4, no. 213, 2008.

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

[5] S. Kimura, K. Ide, A. Kashihara, M. Kano, M. Hatakeyama, R. Masui, N. Nakagawa, S. Yokoyama, S. Kuramitsu, and A. Konagaya, "Inference of S-system models of genetic networks using a cooperative coevolutionary algorithm," *Bioinformatics*, vol. 21, no. 7, pp. 1154–1163, 2005.

[6] N. Noman and H. Iba, "Reverse engineering genetic networks using evolutionary computation," *Genome Informatics Series*, vol. 16, no. 2, pp. 205–214, 2005.

[7] Y. Maki, T. Ueda, M. Okamoto, N. Uematsu, K. Inamura, K. Uchida, Y. Takahashi, and Y. Eguchi, "Inference of genetic network using the expression profile time course data of mouse P19 cells," *Genome Informatics*, pp. 382–383, 2002.

[8] J. Vohradský, "Neural network model of gene expression," *The FASEB Journal*, vol. 15, no. 3, p. 846–854, 2001.

[9] M. A. Savageau, *Biochemical systems analysis: A study of function and design in molecular biology*. Reading, MA: Addison-Wesley, 1976.

[10] P. Mendes, W. Sha, and K. Ye, "Artificial gene networks for objective comparison of analysis algorithms," *Bioinformatics*, vol. 19, no. Suppl 2, pp. ii122–ii129, 2003.

[11] S. Kikuchi, D. Tominaga, M. Arita, K. Takahashi, and M. Tomita, "Dynamic modeling of genetic networks using genetic algorithm and S-system," *Bioinformatics*, vol. 19, no. 5, pp. 643–650, 2003.

[12] N. Noman, L. Palafox, and H. Iba, "Reconstruction of gene regulatory networks from gene expression data using decoupled recurrent neural network model," in *Natural Computing and Beyond*. Springer, 2013, pp. 93–103.

[13] A. Mizeranschi, N. Kennedy, P. Thompson, H. Zheng, and W. Dubitzky, "The influence of network topology on reverse-engineering of gene-regulatory networks," *Procedia Computer Science*, vol. 29, pp. 410–421, 2014.

[14] D. Tominaga, N. Koga, and M. Okamoto, "Efficient numerical optimization algorithm based on genetic algorithm for inverse problem," in *Proc. of Genetic and Evolutionary Computation Conference*. Morgan-Kaufmann, 2000, pp. 251–258.

[15] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. of IEEE International Conference on Neural Networks*, vol. IV, 1995, pp. 1942–1948.

[16] F. M. Lopes, R. M. Cesar, and L. D. F. Costa, "Gene expression complex networks: Synthesis, identification, and analysis," *Journal of Computational Biology*, vol. 18, no. 10, pp. 1353–1367, 2011.

[17] T. Pramila, W. Wu, S. Miles, W. S. Noble, and L. L. Breeden, "The Forkhead transcription factor Hcm1 regulates chromosome segregation genes and fills the S-phase gap in the transcriptional circuitry of the cell cycle," *Genes and Development*, vol. 20, no. 16, pp. 2266–2278, 2006.

[18] 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.

[19] M. Böck, S. Ogishima, H. Tanaka, S. Kramer, and L. Kaderali, "Hub-centered gene network reconstruction using automatic relevance determination," *PLoS ONE*, vol. 7, no. 5, p. e35077, 2012.

[20] J. Mazur, D. Ritter, G. Reinelt, and L. Kaderali, "Reconstructing nonlinear dynamic models of gene regulation using stochastic sampling," *BMC Bioinformatics*, vol. 10, no. 448, 2009.

[21] J. Borgdorff, M. Ben Belgacem, C. Bona-Casas, L. Fazendeiro, D. Groen, O. Hoenen, A. Mizeranschi, J. L. Suter, D. Coster, P. V. Coveney, W. Dubitzky, A. G. Hoekstra, P. Strand, and B. Chopard, "Performance of distributed multiscale simulations," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 372, no. 2021, 2014.

[22] M. Mendoza, "Exploring ensemble learning techniques to optimize the reverse engineering of gene regulatory networks," Ph.D. dissertation, Universidade Federal do Rio Grande do Sul, 2014.