, , pp. 1–6

doi: DOI
Submitted

SUBMITTED

Optimal Inference of Asynchronous Boolean Network Models

Guy Karlebach[1,*]

[1]Computer Science, Fitchburg State university, Pearl St, 01420, MA, USA ORCID 0009-0008-8856-1553
[*]Corresponding author. gkarlebach
Keywords:Boolean Network, Asynchronous Dynamics, Psueodtime
FOR PUBLISHER ONLY Received on Date Month Year; revised on Date Month Year; accepted on Date Month Year

Abstract

Associations between phenotype and genomic and epigenomic markers are often derived by correlation. Systems Biology aims to make more robust connections and uncover broader insights by modeling the cellular mechanisms that produce a phenotype. The question of choosing the modeling methodology is of central importance. A model that does not capture biological reality closely enough will not explain the system's behavior. At the same time, highly detailed models suffer from computational limitations and are likely to overfit the data. Boolean networks strike a balance between complexity and descriptiveness and thus have received increasing interest. We previously described an algorithm for fitting Boolean networks to high-throughout experimental data that finds the optimal network with respect to the information in a given dataset. In this work, we describe a simple extension that enables the modeling of asynchronous dynamics, i.e. different reaction times for different network nodes. In addition, we present a new method for pseudo-time assignment for single-cell RNA sequencing data that is derived from the modeling procedure. Our approach greatly simplifies the construction of Boolean network models for time-series datasets, where asynchronicity often occurs. We demonstrate our methodology by integrating real data from transcriptomics experiments. These results significantly expand the applicability of the Boolean network model to experimental data.

Key words: Boolean Network, Inference, Gene Expression, Biological Network

Introduction

Cellular processes that alter the functional profile of the cell are subject to complex, adaptable regulation. This regulation is achieved by networks of interactions between molecules that integrate environmental signals and the current state of the cell. In order to understand these networks, various modeling methodologies have been proposed, varying in their degree of complexity and expressiveness[Karlebach and Shamir, 2008]. Due to limited knowledge of the reactions involved in carrying out each regulatory interaction, Boolean models, first introduced by Kauffman, are frequently chosen as a general-purpose modeling methodology. Several questions arise when reconstructing a Boolean network from data. First, as with any computational task, one must ask whether the process can be accomplished efficiently [Karlebach and Shamir, 2012]. The second question is how to find the correct trade-off between model size and the fit to the data. Finally, a method to map continuous or discrete measurements into binary values is needed [**?**]. In addressing the first question, Karlebach and Shamir showed that the problem of optimally fitting a Boolean model to binary data is NP-Complete [Karlebach and Shamir, 2008], making a solution impractical in the worst case and likely challenging for many other instances. The second question is usually addressed by independently minimizing the fit of individual regulatory logic tables to the binarized values of their targets. Bayesian approaches have also been proposed, but these arguably deviate from the simplicity of the Boolean model by introducing a probabilistic parameters. Karlebach and Robinson proposed a non-probabilistic criterion that is based on Kolmogorov Complexity [Kolmogorov, 1968] ,and takes into account both the fit to the data and the total size of the model [Karlebach and Robinson, 2023a], and proposed an algorithm for optimizing according to it. As for the third question, while a gold-standard has not been agreed upon, various methodologies have been developed that generate useful mappings in practice [Berestovsky and Nakhleh, 2013]. An often overlooked challenge in modeling is accounting for variation in reaction times. A gene may be activated at a different rate than another gene that has a regulatory association to it, and this can result in a time-series dataset where not all regulations take effect at every sampling point. A related issue occurs when consecutive sampling times capture the same network state repeatedly. Since the data is usually affected by noise, it may be hard to know when the exact same state has been sampled twice. To address these issues, our goal in this paper is to extend the methodology described in [Karlebach and Robinson, 2023a] for Boolean networks with asynchronous dynamics, thereby obtaining a

rigorous reconstruction methodology that can handle time-series data where the sampling rate has not been optimized, or where the rates of different regulatory interactions differ significantly. In the following section, we specify and extend our methodology and explain its application to asynchronous network modeling. We also show how heuristics that address the computational complexity of the problem can be used in the case of asynchronous dynamics. Finally, we discuss pseudo-time assignment to single-cell RNA-Sequencing experiments that is based on the modeling methodology. In the Results section we describe modeling of real data with asynchronous Boolean networks. In the Conclusion section we summarize our findings and assess the impact of the method on future modeling efforts.

Data and Methods

In this section we will assume that the network reconstruction uses transcriptomics data, although the methodology is applicable to any high-throughput dataset. Additionally, all the values are assumed to have been binarized, i.e. mapped to the set of Boolean values 0 and 1. Mapping of real data to Boolean values is demonstrated in the Results section. A Boolean network can be described by a directed graph G(V,E), where G is a set of nodes or genes, and E are edges such that a gene g's regulators are the nodes from which directed edges extend to g, i.e. $u : (u, g) \in E$. Each set of n regulators of a gene is associated with a logic function that has n Boolean inputs and one Boolean output. The Boolean inputs are determined when the nodes that are associated with the function's inputs are assigned Boolean values. Biologically, the input nodes correspond to regulators, and their combined activity or inactivity determines the Boolean value of their target gene. Given a Boolean assignment to all the nodes in the network, also referred to as a state, the next state can be computed by combining the outputs of all the logic functions. A sequence of states that are derived from one another in this way are called a trajectory. If a state is always followed by an identical state in a trajectory, it is called a steady state. In experimental data, a steady state is usually provided as one measurement per gene or network node. The dynamics of the network are synchronous if the values of all the genes are updated by the logic functions simultaneously. If updates can be delayed for any subset of the genes, for any number of network state traversals, we will say the the dynamics are asynchronous. In gene expression datasets, the logic is often unknown and regulators are at best known approximately. A gene expression dataset consists of a N × M matrix, where N corresponds to the number of genes whose expression level was measured, and M corresponds to the number of experiments. The entry at indices i,j contains a Boolean value that is equal to 1 if the gene is in the active state, and otherwise equal to 0. If the data consists of steady states, every row of the matrix represents a network state in which none of the genes will change their activity without external perturbation. If the data consists of trajectories, sequences of rows correspond to consecutive time points in which the network updates the activity states of some or all of its genes. The data can also consist of both steady states and trajectories. In addition to the gene expression dataset, the modeler determines a set of plausible regulatory interactions. For example, the modeler can identify those genes whose protein products bind to the promoter of a target gene, and refine the selection by intersecting this information with correlations in the expression dataset. From this initial set, the modeler want to choose the optimal one, including the logic tables by which the regulators determine the state of the target. An expression dataset will, in practice, always contain some noise. This means that a single set of regulators taking the same value in two network states will have a different effect on the target in each of these states. An inference methodology must therefore identify where such inconsistencies are a result of experimental noise, and where they are the result of asynchronous dynamics. The approach described in the next section minimizes the sum of such inconsistencies and the number of bits in the network encoding. To briefly state the rationale, every additional network bit doubles the number of possible networks, and similarly every mismatch halves the number of remaining solutions to fit. Hence, a non-random solution will result in a sum that is significantly smaller than the size of the dataset [Karlebach and Robinson, 2023b].

In order to find an optimal solution to the problem, we formulate it as 0/1 Integer programming. First, we assume that the number of regulators of each gene is relatively small, in the sense that the logic table defining the regulation is of manageable size. The variables of the problem are denoted by uppercase English letters. A B variable is defined for every measurement, i.e. an entry in the expression matrix that describes a gene and its activity at a given state, and is equal to 1 if there is a mismatch between the observed value of the gene at that measurement and the value that the model assigns it. An I variable is defined for every combination of regulator values, and is equal to 1 if the state of the target gene is set to 1 for that combination, and otherwise it is equal to 0. An R variable is defined for every potential regular-target pair. It is equal to 1 if the regulator is chosen for the optimal model, and otherwise to 0. A V variable is defined for every gene and possible number of regulators for that gene, and is equal to 1 if the gene has at least that number of regulators, and otherwise it is equal to 0. At the end of the section we will also define a D variable, which will allow us to implement asynchronous dynamics. Using these variables, we first describe the constraints of the model, and then the objective function: Let $C_{i,j}$ denote the observed Boolean value of gene i at experiment j. The corresponding B variable is $B_{g_i,j}$, and it is equal 1 if the value of gene g_i in experiment j does not match the model's assignment, and otherwise 0. If j is the index of a steady state in the data, g_{k+1} is a gene with regulators $g_1, g_2, ..., g_k$, we go over every possible combination of values for these regulators $(w_1, w_2, ..., w_k)$, $w_j \in \{0, 1\}$ and for each one add the following constraint:

$$\sum_{r=1}^{k}(C_{r,j} \cdot (w_r + (1 - 2 \cdot w_r) \cdot B_{g_r,j}) \tag{1}$$

$$+(1 - C_{r,j}) \cdot ((1 - w_r) + (2 \cdot w_r - 1) \cdot B_{g_r,j}))$$

$$+C_{k+1,j} \cdot B_{g_{k+1},j} + (1 - C_{k+1,j}) \cdot (1 - B_{g_{k+1},j})$$

$$< (2 - I(w_1, w_2, ..., w_k)) \cdot (k + 1)$$

where $I(w_1, w_2, ..., w_k)$ is the output of the Boolean function that determines the value of g_{k+1}. This constraint means that if the output variable $I(w_1, w_2, ..., w_k)$ was set to 1, whenever the combination $w_1, w_2, ..., w_k$ appears, the output (the value of g_{k+1}) must be 1. If the data contains trajectories, the observed values of the target gene and the corresponding 0/1 IP variables will be taken from the subsequent time point, at which the regulation is expected to take effect if the model is synchronous.

Similarly, we add the following constraint to account for the case where $I(w_1, w_2, ..., w_k)$ is set to 0,:

$$\sum_{r=1}^{k}(C_{r,j} \cdot (w_r + (1 - 2 \cdot w_r) \cdot B_{g_r,j}) \quad (2)$$

$$+(1 - C_{rj}) \cdot ((1 - w_r) + (2 \cdot w_r - 1) \cdot B_{g_r,j}))$$

$$+C_{k+1,j} \cdot (1 - B_{g_{k+1},j}) + (1 - C_{k+1,j}) \cdot B_{g_{k+1},j}$$

$$< (I(w_1, w_2, ..., w_k) + 1) \cdot (k + 1)$$

Next, for every gene g_i and each one of its regulators g_j, we create a Boolean variable R_{ij}. In other words, every potential regulator of gene g_i is associated with an R variable for that gene. For every two different assignment of values to g_i's regulators, i.e. inputs to the logic function that sets the value of g_i, the sum of R variables of regulators which have different values in the two assignments is constrained to be greater than the differences between the I variables that determine the outputs for these assignments. For example, with two regulators R_1 and R_2 and two assignments 01 and 00 to the variables respectively, R_2 must be greater than $I_{01} - I_{00}$ and also greater than $I_{00} - I_{01}$. If the outputs for these two assignments are different, only the change in R_2 can explain this difference,as R_1 has the same value in both assignments. More generally, this constraint means that two different outputs can never occur for the exact combination of regulatory inputs, for otherwise the regulatory logic is not a function. The V_{ik} variable, which is defined for gene i and every possible number of regulators k of that gene, is constrained to be greater than $\frac{0.5}{indegree(g_i)}$ if $k = 1$ or $(\frac{i-1}{indegree(g_i)})$ if $k > 1$. Now we can set the weights of variables in the objective to match the inference criterion: First, a weight of 1 is given to B variables. Now if r regulators are chosen for a gene, all its V variables $1..r$ will be set to 1. Therefore, we set the weight of the first R variable of the gene to be the log base 2 of the number of ways to choose a first regulator plus the log base 2 of the number of logic tables possible for one regulator. We then set the weight of the second R variable of the gene to be the log base 2 of the number of ways to choose a second regulator after the first one was already chosen plus the log base 2 of the number of logic tables with two regulators, minus the log base 2 of the number of logic table with one regulator. So the cost of encoding the logic tables cancel out by consecutive V variables, while the cost of choosing the regulators is produced by the combination of all the V that are set to 1. If we denote the number of logic tables with k regulators as L_k, the weight of the k^{th} V variable is set to $log_2(L_k) - log_2(L_{k-1}) + log_2(\frac{N-k+1}{k})$ So far, we assumed that the updates of the model are synchronous. We now adapt the 0/1 IP formulation to fit asynchronous dynamics, by modifying how it models trajectories. We add a new type of variable called the D variable. This variable is defined for every constraint that involves the B variables in a trajectory, as defined in (1) and (2). It is added to the right hand side of the constraint, and therefore if it is equal to 1 it allows the output of the logic function to not agree with its inputs. We further constrain the D variable to be smaller than 1 minus the differences between the chosen value of target gene at the state at which the regulatory effect is taking effect and the previous state, i.e., the value selected for the gene by the model at these states. This constraint only allows the output of the logic function to disagree with its input if the output does not change, i.e. if the regulatory update is not immediate. Using the same notation as before, the additional constraints on the D variable can be described as follows:

$$1 - (C_{k+1,j+1} * (1 - B_{g_{k+1,j+1}}) + (1 - C_{k+1,j+1}) * B_{g_{k+1,j+1}}$$

$$-(C_{k+1,j} * (1 - B_{g_{k+1,j}}) + (1 - C_{k+1,j}) * B_{g_{k+1,j}})) >= D \quad (3)$$

$$1 - (C_{k+1,j} * (1 - B_{g_{k+1,j}}) + (1 - C_{k+1,j}) * B_{g_{k+1,j}}$$

$$-(C_{k+1,j+1} * (1 - B_{g_{k+1,j+1}}) + (1 - C_{k+1,j+1}) * B_{g_{k+1,j+1}})) >= D \quad (4)$$

We set the weight of every D variable to 1 in the objective function. Consider a network M that is the optimal solution for some dataset T. If it sets the value of the D variable to 1 at some time t, then the corresponding target must exert a regulatory change on one of its own targets after the delay introduced by the D variable, for otherwise a better solution could have been obtained without setting the D variable. Therefore, the trajectory of the model when there is no delay (i.e. when the D variable is not set) is different than the one it uses in the optimal solution. Now if we set the D variable to 0 instead, set the suffix of the trajectory from time t to fit exactly the trajectory of the model that was fit to T from time t and afterwards, then M must also be optimal for this new trajectory T'. If not, and there is another better fit model, then when flipping the bits back, it will still be better than M on the original dataset, whether M uses the D variable or not. Therefore, like the B variables, every D variable is equivalent to one bit in the encoding of the network. As is the case for any 0/1 Integer Programming formulation, the value of the objective function is a sum of the weights of variables that are set to 1 in the solution. Finally, for each target gene we constrain the first D variable in each trajectory to be smaller or equal to the sum of the target's R variables, such that the target would only be able to use the D variables if it has regulators assigned to it. Powerful solvers like Gurobi have dramatically improved our ability to solve 0/1 Integer Programming problems. However, custom heuristics offer even better performance than those devised for the general problem. We now describe such heuristics.

Perhaps the simplest heuristic for a trajectory is to perform a single pass over the data, state by state starting from the first state, and to record every input-output pair observed as long as it does not conflict with pairs observed before it. When a conflict happens, the value of the target gene is flipped to match the output that was previously observed. A more sophisticated approach was suggested by Karlebach and Robinson [Karlebach and Robinson, 2023a], and can be applied to an expression data sets composed of either steady states or equal-length trajectories:

1.Choose a set of regulators.

2.If the set has a single steady state, return it as a solution.

3.If the set has s single trajectory, solve any inconsistencies using the single-pass heuristic, and return it as a solution.

4.If the size of the set of steady states or trajectories is larger than 1 but the set is consistent with the regulators, return that set of states as a solution, possibly removing some redundant regulators by backward elimination.

5.Otherwise, cluster the states and round the cluster centers into Boolean vectors, then solve the problem recursively for the cluster centers. This generates a set of consistent states S. For

every state in the original set, choose its closest neighbor in S, and flip its values one by one to match the neighbor's values until all inconsistencies with states in S have been resolved, or until it is equal to the neighbor, which is already consistent. At that point add it to S so it can be compared to states that have not been made consistent yet. At the end of the process, return S excluding the cluster centers.

The set of regulators in step 1 can be chosen from the current LP solution, for example all regulators which have a value of at least 0.5. The number of clusters should be such that the clusters contain either one state or more than two, since a cluster center for two Boolean states may contain multiple values of 0.5 whose rounding to 0 or 1 is arbitrary. If the dataset contains both steady states and trajectories, then the recursive heuristic can be run for the steady states, and then the resulting logic can be used to remove inconsistencies from the trajectory using the single-pass heuristic. If trajectories have different lengths, equal-sized contiguous subsequences of trajectories can be solved by the recursive heuristic, and the remaining inconsistencies then resolved by the single-pass heuristic. Care should be taken that clustering of these subsequences is biologically meaningful, for otherwise poor solutions may be result due to their incompatibility.

To complete the modeling methodology, the heuristic needs to be adapted for asynchronous dynamics. If a gene's value does not match the output expected by the values of its regulators, but it is consistent with the value of the gene in the previous time point, then it is no longer flagged as an inconsistency. Additionally, when fixing inconsistencies by performing a pass over trajectories and building a set of logic functions, functions are only updated when their target genes change their values between consecutive time steps. With these changes, the heuristic can apply to asynchronous trajectories, or a combination of steady states and such trajectories.

We have integrated this heuristic into our 0/1 Integer Programming implementation in order to improve the upper bound on the nascent solution.

Psuedo-time assignment, also known as trajectory inference, is the ordering of Single-cell RNA Sequencing experiments into trajectories and steady states. We have developed a novel pesudo-time assignment algorithm that is based on our modeling methodology, and is analogous an expectation-maximization procedure. Initially, the reconstructed network is set to the model that contains all the candidate edges, with some initial logic. For example, if edge signs are available, a rule like "inhibitors win over activators' can be used. Next, using the initial network, network trajectories from the states corresponding to the observed Boolean states of the cells are generated: each trajectory is extended until an attractor is reached, and also includes all attractor states. Each Boolean cell state is then mapped to its closest state in the trajectories that were obtained. Contiguous sequences of states are assigned to the corresponding trajectories time points, where trajectories of length 1 are set as steady states. Using the new assignment of pseudo-time, a new network is reconstructed from the data. The process repeats itself until the value of the solution stops improving. In the EM analogy, the latent variables are the cell assignments to pseudo-time points, and the network edges and logic are the model parameters.

In the next section we demonstrate the effectiveness of our method on two real datasets.

Results

The methodology described above was implemented using the Gurobi python API, which allows for a high level of correspondence between the code and the mathematical description of the model. We set the solver parameter OBBT to the value 3. The straightforward correspondence between the python API and the variables and constraints of the problem will enable other programmers an easy access to our implementation.

To test our method on real experimental data, we obtained the microarray dataset GSE49650 of synchronized yeast cells from the Gene Expression Omnibus. Preprocessing of the .CEL files was done using the rma function in the Bioconductor package affy, using default argument values. Additionally, every trajectory (time-series) of every gene was smoothed using the R functions smooth.spline with smooth parameter 0.5 and then approxfun (in the stats package) with default argument values. The x-coordinate values for the smoothing were the times at which the measurements were taken in minutes, and the y-coordinate values were the array intensities. We used the BASCA method as implemented in the R package Binarize [Hopfensitz et al., 2012], with default arguments, for mapping from continuous to Boolean values. Every trajectory was binarized separately. We used the yeast cell cycle model from Cho et al. [Cho et al., 2019], with the edges as the candidate regulatory connections. Complexes were modeled using the expression of one of their genes. After fitting the model, the percentage of mismatches was 16.64 %.

In Cho et al., edges were associated with activation or repression activity. Compared to the inferred logic tables, all the edges agreed on the sign with Cho et al. This result is detailed in Table 1. The inferred logic provided information about combinatorial regulations, i.e. what is the effect of multiple regulators that regulate that target at the same time. This is illustrated in Figure 1 for two of the targets. Interestingly, genes can be separated into those that are active in early stages and those that are active in later stages (Figure 2). Figure 2b shows an inferred trajectory for one of the time-series, and Figure 2a shows the continuous values that were binarized for two of the genes in the model. It should be noted that not every interaction was inferred. The reasons for this is that either the dataset size is too small, or that the interactions occurring in the dataset do not capture all the interactions in the model. Interestingly, both the synchronous and asynchronous dynamics of the inferred model lead to the same steady state, suggesting a robust design. Perturbing a single gene in the steady state generates an oscillation that settles back to the steady state.

Next, we applied our pseudo-time inference algorithm to the HSC network of Bonzanni et al. [Bonzanni et al., 2013], using GEO dataset GSE75478 which profiles human hematopoeitic stem cells in early differentiation, in order to try to reconstruct the network from single-cell data. The raw data from the file GSE75478_transcriptomics_raw_filtered_I2.csv.gz was preprocessed using the R package scuttle, followed by library-size normalization and log transformation. Binarized values were obtained using the R Binarize package and the kMeans method, with default arguments. As an initial network for the pseudo-time inference algorithm, we used the edge signs published in Bonzanni et al. with the "inhibitors win" rule for combinatorial interactions. The initial rate of mismatches was 0.17, and after convergence it was reduced to 0.09. All the inferred interactions matched the signs of the edges published

Table 1. Inferred Edge Effects : Edge function as appeared in Cho et al. and as inferred from the GSE49650 dataset, for each pair of regulator and target.

Source	Target	Cho etl a.l	GSE49650
MCM1	CLN3	activation	activation
SWI5	CLN3	activation	activation
NRM1	MCM1	repression	repression
CLN3	WHI5	repression	repression
CLN2	CDH1	repression	repression
CLB1	CDH1	repression	repression
SWI4	CLN2	activation	activation
NRM1	SWI4	repression	repression
WHI5	SWI4	repression	repression
CDH1	NRM1	repression	repression
CDH1	CLB1	repression	repression
CDH1	NDD1	repression	repression
NDD1	SWI5	activation	activation

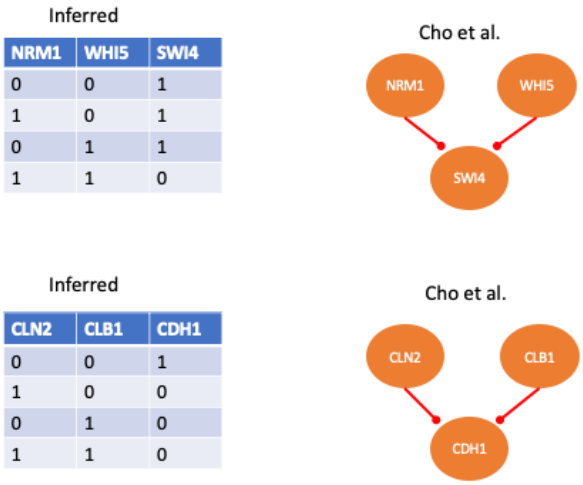

Fig. 1. Regulations in Cho et al [Cho et al., 2019] and inferred combinatorial regulations : Two targets and their regulators from Cho et al., including edge signs, and the combinatorial regulation tables inferred for them.

in Bonzanni et al., though the number of inferred interactions was smaller, likely due to the noisy nature of scRNA-Seq data. We conclude from both experiment that accurate network reconstruction is possible from varying data types and in the presence of asynchronous dynamics.

Conclusion

In this work we have presented a methodology that chooses an optimal model given a time-series dataset that contains asynchronous updates. Our methodology, implemented in software and publicly available for the community, can greatly enhance researchers' ability to understand their data in the context of a regulatory network. The experimental datasets that we analyzed exhibit asynchronicity, but nevertheless our algorithm was able to successfully infer the regulatory

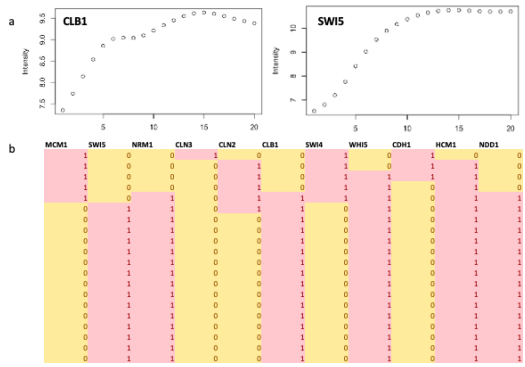

Fig. 2. Inferred Trajectory and continuous values for two of the genes : continuous values used as input for the network reconstruction, and one of the reconstructed trajectories after resolving all inconsistencies with the inferred network.

interactions. We were also able to utilize the inference method in a pseudo-time inference algorithm that makes use of network trajectories. Based on these results, we believe that the algorithm is applicable to a broad range of high-throughput datasets. Several challenges are left for future work: first, the computational efficiency of the method should be further improved, as the problem it addresses is likely to present a variety challenging instances. Another challenge is identifying the best approach for selecting a gene's candidate regulators out of the total set of network nodes. Finally, the topic of binarization of continuous or discrete data into Boolean values deserves more study. Such work should examine the effects of different binarization techniques on inference of asynchronous dynamics systematically, and their synergy with different experimental data types and preprocessing procedures. Special attention should be given to the effects of different technology, particularly single-cell vs. bulk RNA-Seq.

Conflict of Interests

The authors declare that they have no conflicting interests.

Availability of data and software code

The code and data are publicly available on GitHub: https://github.com/karleg/MEDSI

References

N. Berestovsky and L. Nakhleh. An evaluation of methods for inferring boolean networks from time-series data. *PLoS ONE*, 8(6), 2013. doi: doi:10.1371/journal.pone.0066031.

N. Bonzanni, A. Garg, K. A. Feenstra, J. Schutte, S. Kinston, D. M. Saavedra, J. Heringa, I. Xenarios5, and B. Gottgens. Hard-wired heterogeneity in blood stem cells revealed using a dynamic regulatory network model. *Bioinformatics*, 29: i80–i88, 2013. doi: 10.1093/bioinformatics/btt243.

C. Cho, C. Kelliher, and S. Haase. The cell-cycle transcriptional network generates and transmits a pulse of transcription once each cell cycle. *Cell Cycle*, 18(4):363–378, 2019. doi: https://doi.org/10.1080/15384101.2019.1570655.

M. Hopfensitz, C. Muessel, C. Wawra, M. Maucher, M. Kuehl, H. Neumann, and H. A. Kestler. Multiscale binarization of gene expression data for reconstructing boolean networks.

IEEE/ACM Transactions on Computational Biology and Bioinformatics, 9(2):487–498, 2012.

G. Karlebach and P. N. Robinson. Computing minimal boolean models of gene regulatory networks. *Journal of Computational Biology*, 30:2–22, 2023a. doi: DOI:10.1089/cmb.2023.0122.

G. Karlebach and P. N. Robinson. Computing minimal boolean models of gene regulatory networks. *Engrxiv*, 2023b. doi: https://doi.org/10.31224/2566.

G. Karlebach and R. Shamir. Modelling and analysis of gene regulatory networks. *Nature Reviews Molecular Cell Biology*, 9(10):770–780, Sept. 2008. doi: 10.1038/nrm2503. URL `https://doi.org/10.1038/nrm2503`.

G. Karlebach and R. Shamir. Constructing logical models of gene regulatory networks by integrating transcription factor–DNA interactions with expression data: An entropy-based approach. *Journal of Computational Biology*, 19 (1):30–41, Jan. 2012. doi: 10.1089/cmb.2011.0100. URL `https://doi.org/10.1089/cmb.2011.0100`.

A. Kolmogorov. Logical basis for information theory and probability theory. *IEEE Trans. Inf. Theory*, IT-14: 662–664, 1968.