Inferring Probabilistic Boolean Networks from Steady-State Gene Data Samples

Vytenis Šliogeris[1], Leandros Maglaras[2], and Sotiris Moschoyiannis[1(✉)]

[1] School of Computer Science and Engineering, University of Surrey, Guildford, UK
v.sliogeris@surrey.ac.uk, s.moschoyiannis@surrey.ac.uk
[2] School of Computer Science and Informatics, De Montfort University, Leicester, UK
leandros.maglaras@dmu.ac.uk

Abstract. Probabilistic Boolean Networks have been proposed for estimating the behaviour of dynamical systems as they combine rule-based modelling with uncertainty principles. Inferring PBNs directly from gene data is challenging however, especially when data is costly to collect and/or noisy, e.g., in the case of gene expression profile data. In this paper, we present a reproducible method for inferring PBNs directly from *real* gene expression data measurements taken when the system was at a steady state. The steady-state dynamics of PBNs is of special interest in the analysis of biological machinery. The proposed approach does not rely on reconstructing the state evolution of the network, which is computationally intractable for larger networks. We demonstrate the method on samples of real gene expression profiling data from a well-known study on metastatic melanoma. The pipeline is implemented using Python and we make it publicly available.

Keywords: Steady-state data samples · Network structure · Dynamics · Discretisation · Predictor sets · Perceptron · Complex networks

1 Introduction

Rapid progress in the development of next-generation sequencing technologies for genomics has provided valuable insights into complex biological systems [12]. Modelling single-cell or gene networks is becoming increasingly important. The question of modelling complex molecular regulatory networks is an important one for bioinformatics. The goal of systems biology is to intervene on the state of the cell, using the dynamics of the underlying regulatory network. A model that could accurately represent such dynamics could be used for analysis, including control [14,19,26,27,36], steady-state distribution [8,18,24,31], observability [28, 37,38]. Such analyses aid the development of genetic therapies [11].

Boolean Networks (BNs) were introduced for this purpose by Kauffman [15]. In brief, a BN comprises a set of Boolean variables, each variable representing the on/off state of a gene, while interactions between genes are expressed by Boolean functions. It was found that even randomly generated BNs exhibit behaviour reminiscent of gene regulatory networks, with naturally arising attractor states which represent cell types or the phenotype [6,35]. This explains the popularity of BNs for modelling gene interactions [2,10].

However, with few exceptions, gene expression data suggests a number of possible successor states to any given state in a BN, thereby refuting the determinism inherent in BNs. Thus, a *probabilistic* BN (PBN) was introduced by Shmulevich et al. [30] in which the definition of a BN was adapted such that for each gene, at each time point, a Boolean function (and predictor gene set) is chosen with some conditional probability [29].

Inferring the PBN representation of a gene regulatory network (GRN) is quite involved. First, the directed graph expressing interactions between genes needs to be constructed; then, the Boolean functions need to be determined; followed by determining the probabilities of selecting a Boolean function as well as the number of candidate functions on each gene. Existing work (cf Sect. 2) tends to focus on inference from *time-series* gene expression data as the temporal aspect reveals the transition structure of the corresponding PBN. However, as already pointed out in [4], there are concerns over the number of (typically expensive to obtain) observations needed in such gene microarray data. Approaches based on ODEs (e.g., [21]) require lots of observations to tune the large number of parameters of the model, while in practice only a handful are available. More such observations are available when the underlying gene network is at a *steady state* [31], e.g, see gene expression profiles of melanoma by Bittner et al. [5].

In this paper, we propose a systematic method for inferring PBNs directly from real gene expression data measurements, collected using microarray technology, when the system is at a *steady-state.* The steady-state (long-run) behaviour of a PBN is of interest to system biology as it allows to determine the long-term influence of a gene on another gene or determine the long-term joint probabilistic behaviour of a few selected genes [31].

The key contribution of our paper is a reproducible pipeline for going from gene (steady-state) data samples to the PBN representation of the long-run behaviour of the underlying genetic network. We use a predictor gene set rather than temporal data to infer the "transition structure". Unlike other proposals, our method does not require the construction of the *probability transition matrix*, whose size grows exponentially on the number of nodes, and hence becomes computationally intractable for larger networks [1].

The remainder of the paper is structured as follows. Section 2 outlines related work. Preliminary background knowledge is presented in Sect. 3. The main algorithm for our inference method is in Sect. 4. PBNs are produced in Sect. 6 using the process described in Sect. 5. Concluding remarks are in Sect. 7.

2   Related Work

There have been various methods for PBN inference, focusing on causality, using different types of gene data [13]. Previous work on PBN inference from time series gene data includes [32], SCODE [21] with ODEs, and most recently the Stochastic Conjunctive Normal Form (SCNF) -based method by Apostolopoulou et al. [3] which can address larger networks.

Previous work on inference from steady-state data samples is relatively limited and goes back to Shmulevich et al. [31]. A tool for computing the steady-state distribution (ssd) probabilities has been proposed in [23]. Melkman et al. [22] infer *threshold* PBNs, a particular version of PBNs where every input threshold function of a node must have the same number of parameters and also satisfy certain stringent conditions. Kobayashi et al. [18] construct PBNs from BNs by casting inference as an integer linear programming problem and construct a PBN that fits the given steady-state distribution.

Kim et al. [17] use steady-state gene data samples from the study on *metastatic melanoma* by Bittner et al. [5] (we use the same data here). They choose the genes for their PBN using a combination of Coefficient of Determination (COD) analysis and biological background knowledge (we do not assume any prior knowledge). For the functions, they ternarise their data, and construct Lookup Tables in place of the functions for each gene. They also analyse the PBNs produced by analysing the steady-state distribution (ssd) of the resulting network.

Shmulevich et al. [30], who introduced PBNs, describe a method for determining functions for nodes in a PBN. This requires finding sets of input genes which have high COD with the target gene, and using the predictive model used for the calculation of the COD as the function for the particular set of input genes. The probability for choosing the particular input gene set is proportional to the COD of the input gene set.

Discretisation of gene data is an important factor for inference. Chen et al. [7] describe a method for quantising gene data using the expressions of *housekeeping* genes within the dataset. Housekeeping genes are genes which keep a constant expression, as they perform important functions within the cell. Since they have a constant expression, they can be used to estimate the probability distribution function (PDF) of the gene expressions within a microarray. The constructed PDF can be used for using a hypothesis test to determine whether or not a gene is over- or under-expressed. However, this method hinges on knowledge of which of the genes are housekeeping genes and this typically is not readily available.

As discussed in the introductory section, we focus on constructing PBNs from real, microarray gene data samples, collected while the system is in a steady-state, instead of simulated, time-series data or starting from BNs. We present a reproducible method to perform such a task.

3   Preliminaries

3.1   Boolean Networks

A BN [15] is a directed graph, $G = \{V, E\}$, comprised of vertices V and edges E. The vertices $v \in V$ represent the Boolean variables, which in this case represent genes in a gene regulatory network. The directed edges $\{v_i, v_j\} = e_{i,j} \in E$ represent that one variable, v_i, influences another, v_j. Each vertex is associated with a Boolean function f_i given by $f_i : \{0, 1\}^{n_{in}} \mapsto \{0, 1\}$. The input for f_i is a Boolean vector of length n_{in}, which represents the states of all of the input vertices, and the output is a single Boolean value, which is then used as the next state of the variable v_i. For a vertex i, the *input* vertices are the vertices from which all incoming edges originate, given by $\{v_j | \exists \{v_j, v_i\}\} = e_{j,i} \in E$.

3.2   Probabilistic Boolean Networks

Probabilistic Boolean networks are an extension of Boolean networks. They are directed graphs G, as in Boolean networks, except each function f_i for each node i in the case of Boolean networks is replaced by a set of Boolean functions $F_i = \{f_i^1, f_i^2, \ldots, f_i^{l_i}\}$, and probabilities $c_i = \{c_i^1, c_i^2, \ldots, c_i^{l_i}\}$. Hence, the logical function f_i has l_i possibilities, each with a corresponding conditional probability of being selected at every time step.

More formally, during run time, a function f_i^j for the node v_i is chosen with probability c_i^j, $j \in [1, l_i]$. PBNs are an extension to BNs in the sense that if each node within a PBN has a single function, it becomes identical to the BN.

3.3   State Transition Graphs

For each PBN there exists a state transition graph (STG). An STG is a directed graph $G = \{V, E\}$, where the vertices $v_i \in V$ represent the possible states of the PBN, and the edges $\{v_i, v_j\} = e_{i,j} \in E$ represent the possibility of a transition from state v_i to v_j. Since the probability of getting to another state v_j only depends on the current state v_i, we can say that the STG is a Markov chain.

By saying that the PBN has a steady state distribution (ssd), we mean that the STG of the PBN has a steady state distribution. For an STG to have an SSD, it needs to be *ergodic* - that is, every state can be reached from every other state. To guarantee that the STG is ergodic, random perturbations with low probability are introduced to the PBN.

3.4   Microarray Gene Data Samples

The data used to infer a PBN in our work was taken from the study of metastatic melanoma found in Bittner et al. [5], which has been extensively studied in the literature [17, 25, 27, 33]. The study extracts and analyses the gene expression profiles of 31 melanoma cells using microarray technology. To make sure that the gene expression levels used in inferring the corresponding PBN are those

of genes when the network is at a steady state, the Kolmogorov-Smirnov (KS) statistic is applied, as discussed in more detail in Sect. 5.

To utilise a particular gene in DNA, see [7], assuming the cell is at a steady-state, the relevant segment of the molecule must first be transcribed, producing messenger RNA (mRNA) which is accessible to the rest of the proteins. The quantity of mRNA in a cell signifies the degree of protein production associated with a particular gene.

DNA microarrays measure the presence of mRNA within a cell. The microarrays consist of a surface with an array of robotically placed complementary DNA for the genes to be analysed. mRNA tightly bonds with complementary DNA, hence the microarray can be used to isolate different mRNA molecules. The process is known as *hybridisation*.

The quantity of mRNA within a cell is measured by tagging the mRNA with fluorescent molecules, hybridising them with a microarray, and exciting the fluorescent molecules. The emitted brightness is proportional to the amount of mRNA present.

Since the amount of mRNA differs depending on the gene, the data is normalised by dividing the values recorded by the values recorded from a reference probe. Since values recorded are non-negative, the ratio values are in the range of $[0, \infty)$. Furthermore, since we would expect the values of within the reference probe and the sample to not be different, the median for the ratio values is expected to be 1. These are the values provided by Bittner et al. [5] in the form of a matrix of size 8,150 (number of genes) by 31 (number of samples). A small sample of the raw data is shown later in Fig. 1(a).

For demonstrating our method of inferring a PBN, we work with the subset of melanoma genes analysed by Datta et al. [9], which are extensively studied in the literature [17,18,25,27,33], namely *WNT5A, pirin, S100P, RET1, MART1, HADHB* and *STC2*. This offers straightforward validation for our approach since it produces the same PBN.

It is worth noting that larger PBNs may be constructed following the pipeline described in this paper, and we have constructed the 28 node PBN given in [33] as well as a 70 node PBN, which includes the 28 nodes already studied in [33] padded with the 42 nodes with the highest weighting of importance, using discriminative weights [5], which determine how a gene changes during the experiment compared to the control cells.

3.5 Coefficient of Determination

Coefficients of Determination (CODs) were described by Kim et al. [16] as a method to determine which gene determines the state of which other gene. A COD of a target variable, Y, with regards to an input variable, X, is a measure on how well the target variable can be predicted using the input variable. A predictive model f is used to predict the value of the target variable with and without the input variable, and compute the errors $\bar{e}$ and e respectively. The relative change of error of the predictive model is the COD θ, given by Eq. 1:

$$\theta = \frac{\bar{e} - e}{\bar{e}} \tag{1}$$

There are no constraints on what can be used as a predictive model. We opted for a perceptron. This is because there exists a closed-form solution for linear regression of the perceptron, described by Kim et al. [16], which can be used instead of training. This aids in lowering the computation time.

The weights of a perceptron, A, can be computed using the closed form solution:

$$
\begin{aligned}
A &= R^+ \cdot C \\
R &= X \cdot X^T \\
C &= X \cdot Y
\end{aligned}
\tag{2}
$$

3.6   Discretisation

Since PBNs use discrete values, the gene data which consists of real values has to be discretised. Discretisation is a process where values get mapped from the real value domain to the integer domain. For the problem at hand, since genes can be in one of two states, the range of the function should be either 0 or 1. Hence the function should take the form of:

$$f : G \to G_d, x \geq 0, \forall x \in G, y \in \{0, 1\} \forall y \in G_d \tag{3}$$

Such a method is described in detail in [34]. It consists of deciding upon a threshold value t with which all real values are compared. Each value then gets mapped to 0 if it is below the threshold, and to 1 otherwise, as given by Eq. 4.

$$G_d[x, y] = \begin{cases} 0 & G[x, y] < t \\ 1 & G[x, y] \geq t \end{cases} \tag{4}$$

The threshold may be any metric. Common metrics are means or medians. The threshold may also be the boundary between the top $x\%$ of entries and the rest.

Shmulevich et al. [29] describe a process of using k-means clustering to cluster the data, and assigning values to the data points depending on the cluster they belong to. However, since half of the data points lie in the range $(0, 1)$, and the other half is in the range $(1, \infty)$, the lower cluster ends up larger, resulting in a larger threshold that produces more zeros. This can be remedied by performing k-means clustering on the logarithms of the data points. This makes the ranges of both halves the same, producing more representative clusters.

4   Inference of PBNs

In this section we describe the inference method and how it can be implemented. Our approach to inferring a PBN starts with the real gene expression data in the form of a matrix G as input (see Fig. 1(a)), and produces a PBN (see Fig.

1(b)). The input matrix is of size $m \times n$, where m is the number of genes and n is the number of samples.

The method we apply for inferring PBNs draws upon work done by Shmulevich et al. [30]. First, it requires the dataset to be discretised (recall Section 3.6). This process is performed in Algorithm 1.

Data: Input array G, discretisation axis, calculation method
Result: Discretised array G
for *row g in G over axis* **do**
    t = method(G);
    for *element in g* **do**
        if *element ¡ t* **then**
            element = 0;
        else
            element = 1;
        end
    end
    return G ;
end

Algorithm 1: Discretisation algorithm

Given a target gene, n_p sets of genes with the highest CODs are found. This is down following Algorithm 2.

Data: Input array G, number of input genes per tuple k, number of input tuples n
Result: List of predictor-COD-input tuples
tuples = [];
for *targetIndex in 1 . . . G.geneAxisSize* **do**
    indexCombinations =
    generateAllCombinations(1 . . . $G.geneAxisSize \setminus targetIndex$, k);
    // Would return triplets of all possible k-combinations of
        the gene indexes within the array.
    buffer.init(n);
    for *inputCombo in indexCombinations* **do**
        COD, weights = calcCOD(G[targetIndex],G[inputCombo]);
        if *COD > min(buffer.COD)* **then**
            buffer.add(COD, weights, inputCombo);
            buffer.removeSmallestCOD() ;
        end
    end
    tuples[targetIndex] ← buffer;
end
return buffer;

Algorithm 2: Tuple list generator

A buffer of size n_p is initialised, and each possible combination of input genes have their CODs calculated. If a combination of inputs has a COD higher than at least one saved in the buffer, the buffer entry with the lowest COD gets

replaced by the new combination of inputs. This results in a buffer full of input combinations with the highest CODs. One such buffer is initialised per target gene, resulting in n_p input combinations per target gene.

During run-time, a set of input genes is chosen with probability proportional to the COD of the set, and the next state is governed by the state of those input genes in conjunction with the predictive model that was saved. For all intents and purposes, the list with input gene, perceptron weights and probabilities are enough to construct a PBN, as the input genes convey the connectivity, and the perceptron weights convey the logic for that set of input genes. The process is summarised in Algorithm 3.

Data: Input array G, discretisation axis, threshold calculation method, number of input genes per tuple k, number of input tuples n

Result: List of predictor-probability-input tuples

$\hat{G}$ = discretise(G, axis, method);

predictorList = genPredictorList($\hat{G}$, k, n);

CODsum = 0 ;

for *tuple in predictor list* **do**
  | CODsum $\leftarrow$ CODsum + tuple[COD] ;
end

for *tuple in predictor list* **do**
  | tuple[probability] $\leftarrow \frac{tuple[COD]}{CODsum}$;
end

return predictor list;

Algorithm 3: General algorithm for generating predictor lists

Ratio Data for Group of 12 Unclustered Cutaneous Melanomas

UACC-1012	UACC-1529	UACC-647	WM1791-C	UACC-827	HA-A	UACC-930	UACC-903
0.94	1.68	0.99	0.97	3.06	1.83	2.14	0.54
0.58	0.38	1.1	1.03	2.29	1.71	0.72	1.04
0.63	0.2	1.09	1.06	1.01	0.7	0.86	0.87
0.57	0.28	0.83	0.81	0.92	0.97	0.75	1.14
0.73	0.77	1	0.82	0.75	0.57	0.79	0.6
0.66	0.62	0.89	0.58	0.69	1.21	1.14	0.81
0.61	0.51	0.8	0.51	0.56	0.45	0.65	0.78
0.89	0.83	0.96	1.02	1.06	0.19	1.23	0.79
0.88	0.6	0.87	0.8	1	0.52	1.82	1.09
0.58	0.82	1.13	1.1	1.44	0.73	1.89	1.34
2.61	1.85	1.24	6.11	1.4	0.9	2.54	1.24
1.03	0.91	1.07	1.16	0.98	1.07	0.95	0.93
1.19	1.13	1.06	1.17	0.84	0.85	1.01	0.9
1.31	1.58	1.13	1.64	1.41	1.48	1.91	1.39
1.02	0.99	0.78	0.91	0.67	0.78	1.3	0.97
0.52	0.52	0.59	1.63	0.27	0.71	1.5	1.12
0.85	0.82	1.17	2.68	0.59	1.52	1.07	1.1

(a) The input, gene expression values for 31 samples taken at a steady-state, given in the form of a matrix
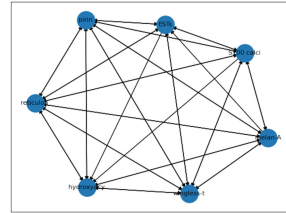
(b) The output is a PBN

Fig. 1. Input and output for the inference method

5   Analysis

The analysis of the generated PBNs in our approach are based on steady-state distribution, which is fairly standard, e.g., see [17]. The PBN is run for T steps

in order to get it within a steady state. Then it is run for the next N steps, recording the state it is at. To confirm whether or not the PBN is in a steady state after T steps, the Kolmogorov-Smirnov (KS) statistic is calculated for the two halves of N.

The entries recorded in N are split in to two halves - one containing states $[0, \frac{N}{2}]$, the other containing $[\frac{N}{2} + 1, N]$. The entries are subsampled with the interval G. The histograms are converted to cumulative distribution functions (CDFs), and the maximum vertical distance between them is found, which is the KS statistic.

The significance test shows the probability of the two CDFs being drawn from the same distribution. If the PBN had not reached a steady state after T steps, the halves of N would be drawn from different distributions, which would be indicated by the KS test. The recorded states are a string of binary values. Therefore, for ease of analysis, they are used as gray-coded integers, and displayed on a histogram (cf. Fig. 2). This makes the horizontal distance on the histogram proportional to the Hamming distance between two network states.

6   Evaluation

We have implemented the pipeline using Python 3 and make it publicly available on https://github.com/UoS-PLCCN/pbn-inference.

We have constructed PBNs of size 7 from data produced by Bittner et al. [5] using different thresholds for the quantisation methods. The thresholds were (a) average of a gene expression; (b) median of a gene expression, and (c) k-means clustering of a gene expression. The data was quantised on a per-gene basis, with each gene having 10 triplets of input genes.

For the construction and validation of the histograms representing the steady-state distribution, we have chosen the parameters to be $T = 10^6$, $N = 4 \cdot 10^6$, $G = 10$ and $R = 100$. On a laptop with 32 GB of RAM and an Intel® Core™ i7-7700HQ Processor, each histogram took around 9 hours to produce. The results are shown in Fig. 2.

(a) Mean   (b) Median   (c) K-Means clustering
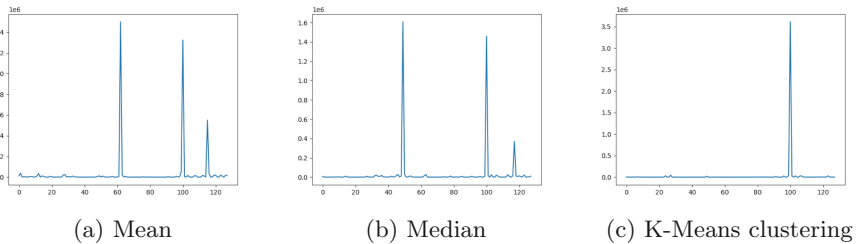
Fig. 2. SSDs of PBNs generated using different quantisation methods. States on the x-axis; SSD probability on the y-axis

It can be seen that the average and the median quantisation methods produce very similar histograms, with three peaks each, and the latter two peaks being

in similar positions. The histogram generated using the PBN constructed from k-means clustering only has one prominent state, which can also be observed in the other two PBNs. It may be constructive to note that the few very prominent states in the histograms shown in Fig. 2 agrees with the assumption claimed by Kim et al. [17] that gene regulatory networks found in nature only occupy a small fraction of the possible state space.

For the purposes of direct comparison, we have trialled the proposed method in the DREAM (Dialogue on Reverse Engineering Assessment and Methods) challenge[1] which offers a benchmark for network inference (DREAM 3) [20] and scored 8th (out of 29).

7   Conclusion

In this work we described the inference a PBN directly from real gene data, collected using microrarray technology, which were taken when the system was at a steady-state. This kind of gene profiling is typically less costly to obtain than time series data, and includes more data points. Using the evaluation methods described in the literature, e.g., by Kim et al. [17], we have concluded that the pipeline works well for the examples provided. However, it is subject to fine-tuning the parameters. We have provided the method in a systematic pipeline which can be reproduced. We made it publicly available on github https://github.com/UoS-PLCCN/pbn-inference.

We note that the method scored 8th (out of 29) in the DREAM challenge and has been used to infer large PBNs (N = 200).

It is worth noting that the proposed method does not require a state transition probability matrix to be produced. It can be extracted from the PBN, however, the time required grows exponentially with the size of the PBN. This means that conventional mathematical methods in the literature that make use of the transition probability matrix may not always be applicable.

One concern is that the transitions get fitted to the quantised dataset. It is widely accepted that the states observed in the dataset are steady states of the cells. Since the transition rules get fitted to the steady states of the cells, the resulting PBN will be driven towards the steady states observed within the data. However, while it is certain that the method captures the long-run behaviour (steady-state) of the underlying gene regulatory network, there is little certainty that the PBN will behave with biological accuracy between the observed steady states. This concern could possibly be addressed by using time-series gene data to augment the method presented here, as this type of data captures the change of gene expression levels with respect to time. This promises to capture the behaviour at and between steady states, without reconstruction of the state evolution of the PBN, and is certainly worth exploring further in future work.

---

[1] https://dreamchallenges.org/project/dream-3-in-silico-network-challenge/.

References

1. Akutsu, T., et al.: Control of Boolean networks: hardness results and algorithms for tree structured networks. J. Theor. Biol. **244**(4), 670–679 (2007)
2. Albert, R., Othmer, H.G.: The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in Drosophila melanogaster. J. Theor. Biol. **223**(1), 1–18 (2003)
3. Apostolopoulou, I., Marculescu, D.: Tractable learning and inference for large-scale probabilistic Boolean networks. IEEE Trans. Neur. Netw. Learn. Syst. **30**(9) (2019)
4. Bar-Joseph, Z.: Analyzing time series gene expression data. Bioinformatics **20**(16), 2493–2503 (2004)
5. Bittner, M., et al.: Molecular classification of cutaneous malignant melanoma by gene expression profiling. Nature **406**, 536–40 (2000)
6. Chatzaroulas, E., Sliogeris, V., Victori, P., Buffa, F.M., Moschoyiannis, S., Bauer, R.: A structural characterisation of the mitogen-activated protein kinase network in cancer. Symmetry **14**(5) (2022)
7. Chen, Y.: Ratio-based decisions and the quantitative analysis of cDNA microarray images. J. Biomed. Opt. **2**(4), 364 (1997)
8. Ching, W.K., Zhang, M.K.N., Akutsu, T.: An approximation method for solving the steady-state probability distribution of probabilistic Boolean networks. Bioinformatics **23**(12), 1511–1518 (2007)
9. Datta, A., Choudhary, A., Bittner, M.L., Dougherty, E.: External control in markovian genetic regulatory networks. Mach. Learn. 4, 52, 3614 – 3619 (2003)
10. Davidich, M., Bornholdt, S.: The transition from differential equations to Boolean networks: a case study in simplifying a regulatory network model. J. Theor. Biol. **255**(3), 269–77 (2008)
11. Fumia, H.F., Martins, M.L.: Boolean network model for cancer pathways: Predicting carcinogenesis and targeted therapy outcomes. PLoS ONE **8**(7), e69008 (2013)
12. Gawad, C., Koh, W., Quake, S.: Single-cell genome sequencing: current state of the science. Nat. Rev. Genet. **17**, 175–188 (2016)
13. Glymour, C., Zhang, K., Spirtes, P.: Review of causal discovery methods based on graphical models. Front. Gene. **10**(524) (2019)
14. Karlsen, M.R., Moschoyiannis, S.: Evolution of control with learning classifier systems. Appl. Netw. Sci. **3**(1), 1–30 (2018)
15. Kauffman, S.A.: Metabolic stability and epigenesis in randomly constructed genetic nets. J. Theor. Biol. **22**(3), 437–467 (1969)
16. Kim, S., Dougherty, E., Bittner, M., Chen, Y., Sivakumar, K., Meltzer, P., Trent, J.: General nonlinear framework for the analysis of gene interaction via multivariate expression arrays. J. Biomed. Opt. **5**, 411–24 (2000)
17. Kim, S., Dougherty, E.R., Chen, Y., Bittner, M., Suh, E.: Can markov chain models mimic biological regulation? J. Biol. Syst. **10** (2003)
18. Kobayashi, K., Hiraishi, K.: Design of probabilistic Boolean networks based on network structure and steady-state probabilities. IEEE Trans. Neur. Netw. Learn. Syst. **28**(8), 1966–1971 (2017)
19. Liu, Y.Y., Slotine, J.J., Barabási, A.L.: Controllability of complex networks. Nature **473**(7346), 167 (2011)
20. Marbach, D., Prill, R.J., Schaffter, T., Mattiussi, C., Floreano, D., Stolovitzky, G.: Revealing strengths and weaknesses of methods for gene network inference. Proc. Nat. Acad. Sci. **107**(14), 6286–6291 (2010)

21. Matsumoto, H., et al.: SCODE: an efficient regulatory network inference algorithm from single-cell RNA-Seq during differentiation. Bioinformatics **33**(15), 2314–2321 (2017)
22. Melkman, A.A., Cheng, X., Ching, W.K., Akutsu, T.: Identifying a probabilistic Boolean threshold network from samples. IEEE Trans. Neur. Netw. Learn. Syst. **29**(4), 869–881 (2018)
23. Mizera, A., Pang, J., Yuan, Q.: Assa-pbn: An approximate steady-state analyser of probabilistic boolean networks. In: Automated Technology for Verification and Analysis. Springer International Publishing, Cham, pp. 214–220 (2015)
24. Moschoyiannis, S., Shields, M.: A set-theoretic framework for component composition. Fundamenta Informaticae **59**(4), 373–396 (2004)
25. Pal, R., Datta, A., Dougherty, E.R.: Optimal infinite-horizon control for probabilistic Boolean networks. IEEE Trans. Sign. Process. **54**(6), 2375–2387 (2006)
26. Papagiannis, G., Moschoyiannis, S.: Learning to control random Boolean networks: A deep reinforcement learning approach. In: Complex Networks 2019, Vol. 881. Springer, Cham, pp. 721–734 (2019)
27. Papagiannis, G., Moschoyiannis, S.: Deep reinforcement learning for control of probabilistic Boolean networks. In: Complex Networks 2020, Vol. 944. Springer, pp. 361–371 (2020)
28. Savvopoulos, S., Moschoyiannis, S.: Impact of removing nodes on the controllability of complex networks. In: Complex Networks (2017)
29. Shmulevich, I., Dougherty, E.R.: Probabilistic Boolean Networks: The Modeling and Control of Gene Regulatory Networks. SIAM (2010)
30. Shmulevich, I., Dougherty, E.R., Kim, S., Zhang, W.: Probabilistic Boolean networks: a rule-based uncertainty model for gene regulatory networks. Bioinformatics **18**(2), 261–74 (2002)
31. Shmulevich, I., et al.: Steady-state analysis of genetic regulatory networks modelled by probabilistic Boolean networks. Comp. Funct. Genom. **4**(6), 601–608 (2003)
32. Silescu, A., Honavar, V.: Temporal Boolean network models of genetic networks and their inference from gene expression time series. Compl. Syst. **13**(2001), 61–78 (2001)
33. Sirin, U., Polat, F., Alhajj, R.: Employing Batch Reinforcement Learning to Control Gene Regulation Without Explicitly Constructing Gene Regulatory Networks, pp. 2042–2048 (2013)
34. Velarde, C., Rubio-Escudero, C., Romero-Zaliz, R.: Boolean networks: a study on microarray data discretization. In: ESTYLF08, Cuencas Mineras (Mieres-Langreo), pp. 17–19 (2008)
35. Voukantsis, D., Kahn, K., Hadley, M., Wilson, R., Buffa, F.M.: Modeling genotypes in their microenvironment to predict single- and multi-cellular behavior. GigaScience **8**(3) (2019). https://doi.org/10.1093/gigascience/giz010
36. Wu, Y., Shen, T.: Policy iteration algorithm for optimal control of stochastic logical dynamical systems. IEEE Trans. Neur. Netw. Learn. Syst. **29**(5), 2031–2036 (2019)
37. Zhang, K., Johansson, K.H.: Efficient verification of observability and reconstructibility for large boolean control networks with special structures. IEEE Trans. Autom. Contr. **65**(12), 5144–5158 (2020)
38. Zhu, Q., Liu, Y., Lu, J., Cao, J.: Controllability and observability of Boolean control networks via sampled-data control. IEEE Trans. Control. Netw. Syst. **6**(4), 1291–1301 (2019)