Chapter 1
Intelligent Learning and Verification of Biological Networks

Helen Richards, Yunge Wang, Tong Si, Hao Zhang, and Haijun Gong

Abstract Machine learning and model checking are two types of intelligent computing techniques that have been widely used to study different complicated systems nowadays. It is well-known that the cellular functions and biological processes are strictly regulated by different biological networks, for example, signaling pathways and gene regulatory networks. The pathogenesis of cancers is associated with the dysfunctions of some regulatory networks or signaling pathways. A comprehensive understanding of the biological networks could identify cellular signatures and uncover hidden pathological mechanisms, and help develop targeted therapies for cancers and other diseases. In order to correctly reconstruct biological networks, statisticians and computer scientists have been motivated to develop many intelligent methods, but it is still a challenging task due to the complexity of the biological system and the curse of dimensionality of the high-dimensional biological data. In this work, we will review different machine learning algorithms and formal verification (model checking) techniques that have been proposed and applied in our previous work and discuss how to integrate these computational methods together to intelligently infer and verify complex biological networks from biological data. The advantages and disadvantages of these methods are also discussed in this work.

1.1 Introduction

Nowadays, targeted therapy has become an important and effective treatment method for many types of cancers, which uses specific chemical compounds to target some mutated genes and proteins implicated in tumorigenesis. To develop targeted therapies, we need to identify important genetic mutations and understand how these

H. Richards
Biomedical Engineering Department, Saint Louis University, St. Louis, MO, USA

Y. Wang · T. Si · H. Zhang · H. Gong (✉)
Department of Mathematics and Statistics, Saint Louis University,
220 N Grand Blvd, St. Louis, MO, USA
e-mail: haijun.gong@slu.edu

T. D. Pham et al. (eds.), *Advances in Artificial Intelligence, Computation, and Data Science*, Computational Biology,
https://doi.org/10.1007/978-3-030-69951-2_1

mutated genes and proteins influence the gene regulatory networks and signaling pathways that are involved in the cancer cell growth. So, understanding of the biological network is one of the key steps to investigate the cellular system and pathogenesis of different diseases. Reconstruction of biological networks involves two major steps: inference and verification, that is, how to intelligently and correctly infer optimal networks, and how to intelligently and automatically verify or falsify the inferred networks. Statisticians and computer scientists have proposed different statistical learning and model checking techniques to implement network inference and verification in the past years.

Different learning methods have been developed to infer the biological networks from high-dimensional omics data generated by the modern genomic technologies. The traditional deterministic methods are based on the ordinary differential equations [3], which is not convenient to handle the noisy high-dimensional data. Most network learning methods are based on the probabilistic graphical models (PGM) [16, 17, 31], for example, the Bayesian network and dynamic Bayesian networks methods. In the PGM, each node represents a variable which could be a gene or protein, and the edge connecting two nodes indicates a possible causal or conditional dependency relationship (which could be activation or inhibition or association). The simplest PGM is a Bayesian network (BN) model, but the BN model can not handle the positive or negative feedback loops which are important processes in the genetic networks and signaling pathways. In most omics datasets, the number of observations or measurements is significantly fewer than the number of genes or proteins, some LASSO-based methods, for example, graphical LASSO [15, 36], have been developed to estimate inverse covariance matrix, which can infer some undirected networks.

To reconstruct a network which contains causality information and allows the feedback loops, different Dynamic Bayesian network (DBN) [17, 31, 32, 37]-based models, which assume a first-order Markov chain, have been proposed to learn directed networks. We are more interested in the regulatory networks which contain more information than the correlation and causality graphs. Most network learning methods can not identify the "activation" and "inhibition" relationship between different nodes in the regulatory networks. Our previous work [22, 35] introduced a signed integer weight by modifying the influence score proposed in [49] to identify the activation/inhibition relationship for each edge on the regulatory networks.

Several biomedical studies [4, 11, 29, 34] found that, in some biological systems, the network structure is time-varying at different stages. For example, the network structure of naive/effector T cells is different from the senescent T cells [11] due to some mutations during tumorigenesis. Recently, time-varying network inference methods were proposed [1, 18, 27, 40, 44], including dynamic vector autoregressive model [18], heterogeneous DBN model [27], L1-regularized logistic regression model [1], and dynamic linear model [44]. However, these methods can only be used to learn time-varying causality networks or undirected graphs, instead of regulatory networks.

There are several steps that are challenging in the reconstruction of time-varying regulatory network than the stationary network inference. The first step before the

network structure learning is to identify when the network structure starts to make changes. This is a change-points detection problem in statistics. The change-point detection is an open problem in high-dimensional time-series statistical data analysis, it is especially difficult if both the number and locations of change-points are unknown. The second challenge is how to learn the optimal time-varying networks at different stages. Our recent work [43] indicates that if we could identify the change-points, we still can use the dynamic Bayesian network model to learn the optimal networks at different stages. Another challenge is how to intelligently and automatically check whether or not the inferred networks are consistent with existing experiments or known databases. It is not realistic to manually verify complex networks which are composed of thousands of genes, proteins, and interactions. Our previous work [19, 20, 22, 24, 25] had introduced and applied different model checking techniques, including statistical model checking, symbolic model checking and probabilistic model checking, to formally analyze the signaling pathway models and gene regulatory networks by checking some desired temporal logic formulas abstracted from the experiment and known database.

Our recent work [43] proposed a preliminary integrative approach to reconstruct time-varying regulatory networks. The reconstruction of regulatory networks from time-series high-dimensional data involves several key steps: identification of change-points, network structure learning and searching at different stages, learn the activation/inhibition relationship for each edge, and verification of inferred networks. In this chapter, we will review several machine learning algorithms and model checking techniques that we have proposed and applied in our previous biological network studies [19, 20, 22, 24, 25, 35, 43] and discuss their limitations in the network inference and verification, and propose a way to integrate these techniques together for the regulatory network reconstruction.

1.2 Statistical Learning of Regulatory Networks

During the pathogenesis of some diseases or stimulation under some conditions, some genetic or protein mutations might change the structure of genetic networks or sequence of signal transduction. For example, the mutations of tumor-suppressor proteins P53 and Rb, and oncoproteins RAS and MDM2 could change the signal transduction in the P53-MDM2, Rb-E2F, and RAS networks. Figure 1.1 illustrates simple regulatory networks, which are composed of three-nodes, with time-varying network structures. These structure changes include some newborn edges connecting two nodes, removal of some old edges, changes of activation/inhibition relationship. To reconstruct a regulatory network, we should first check the network structure is time-invariant or time-varying over time from the high-dimensional time-series data, which is a change-points detection problem in statistics.

To identify the change-points, which describe when the network structure starts to make changes, from the high-dimensional time-series data is a difficult problem in statistics. This is especially challenging if both the number and locations

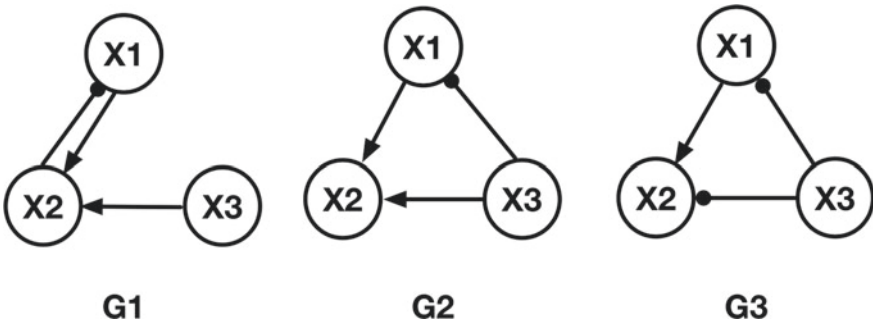

Fig. 1.1 Illustration of time-varying regulatory networks at three different stages. The arrow represents activation and the filled ball arrow represents inhibition relationship

of change-points are jointly unknown. Some unsupervised and supervised learning-based change-points detection methods [13, 14, 26, 30, 39, 41] have been proposed, including the density-ratio test [30], Bayesian inference approaches [13, 41], product partition model [5], reversible jump Markov Chain Monte Carlo sampling algorithm [26], logistic regression [14], and hidden Markov model [39]. If the data is low-dimensional, most of these methods work well. To handle the high-dimensional omics data, most existing methods have their limitations.

1.2.1 INSPECT Change-Points Identification

To choose a change-points identification method that can handle the high-dimensional omics data is the first important step in the network reconstruction, it is challenging if both the number and locations of change-points are unknown. In our recent work [43], we adopted a change-point estimation method called INSPECT proposed by [42] and applied it to identify the change-points from the high-dimensional microarray data. Our studies found this method could be used to identify multiple change-points from high-dimensional time-series data if we can choose some correct values for a small number of parameters.

The INSPECT (informative sparse projection for estimation of change-points) algorithm is based on the ADMM (alternating direction method of multipliers) algorithm to perform sparse singular value decomposition on the CUSUM (cumulative sum) transformation matrix [10, 42] and find sparse leading left singular vectors for the change-points identification. The details are given in [42], for completeness, we summarize some key points of this algorithm in this chapter. Given some high-dimensional time-series microarray data described by a matrix $X = (X_1, \ldots, X_n)^T \in R^{n \times p}$, which consists of p genes measured at n different time points, in which, X_t $(1 \leq t \leq n)$ describes the expression level of all the p genes observed at time t. X_t follows some unknown distribution with a time-changing

mean vector μ_t at different stages. Assuming there are ν change-points denoted by $\{c_1, \ldots, c_\nu\}$ in the data X, where $1 \leq c_1 < c_2 < \cdots < c_\nu \leq n-1$, then, there are $\nu + 1$ stages described by $\{s_1, \ldots, s_{\nu+1}\}$, where $s_i = (c_i, c_{i+1})$. At any stage, the network structure is assumed to be time-invariant, then, the mean vectors $\mu^{(i)}$ is a constant at its stage s_i.

Algorithm 1: Change-points Estimation INSPECT Pseudocode [42]

Input: High-dimensional time-series data D; Parameters (regularization, threshold)

1. Perform CUSUM transformation T, where $[T_{n,p}(X)]_{t,i} = \sqrt{\frac{n}{t(n-t)}}\left(\frac{t}{n}\sum_{j=1}^{n} X_{j,i} - \sum_{j=1}^{t} X_{j,i}\right)$.
2. Find k-sparse leading left singular vectors $\hat{v}_k$ of CUSUM using ADMM algorithm: solve convex optimization problem: $\hat{v}_k = argmax||T^T v||_2$, with the constraint $||\mu^{(i)} - \mu^{(i-1)}||_0 \leq k$, where $i = 1, 2, \ldots, \nu$, and $k \in \{1, \ldots, p\}$.
3. Locate change-points by wild Binary segmentation.

Output: Number and locations of change-points

Algorithm 1 summarizes the procedure of INSPECT method [42] to estimate the number and locations of multiple change-points from high-dimensional time-series data. The first step is to perform the CUSUM (cumulative sum) transformation [10, 42] matrix $T_{n,p} : R^{n \times p} \rightarrow R^{(n-1) \times p}$, which is defined in the Algorithm 1, to obtain the optimal projection direction that are closely aligned with the changes of mean vectors between the stages s_i and s_{i+1}. The second step is to find the k-sparse leading left singular vectors $\hat{v}_k$ [42, 46] of the CUSUM transformation matrix T, which is equivalent to solving a convex optimization problem. Most parts of the network structure should be conserved, so, there should not be too many edge-changes during stage transition. A sparsity is introduced using the constraint $||\mu^{(i)} - \mu^{(i-1)}||_0 \leq k$ to constrain the number of edge-changes. Finally, locate change-points by wild Binary segmentation. More details with proof are in Wang et al.'s work [42].

This work only discussed the INSPECT change-points detection algorithm that we have applied in our previous work, the interested readers could also refer to [5, 13, 14, 26, 39, 41] for different change-points detection methods for comparison.

1.2.2 Network Structure Learning and Searching

Given the time-series dataset D which is expressed by an $n \times p$ matrix showing the expression levels of p genes measured at n different time points. The change-points detection algorithm is implemented first, if there is no change-point, then, the network can be modeled by a stationary dynamic Bayesian network (DBN); if there are at least one change-points, then, the network structure is time-varying, we should infer different DBN models at different stages, that is, time-varying DBNs.

1.2.2.1 Time-Invariant DBN Learning

Dynamic Bayesian Network (DBN) has been reviewed in many previous studies. We will briefly review this method. DBN model is a pair (G, Θ), where $G = (V, E)$ represents a directed graph, in which V represents a set of random variables or nodes, E is a set of edges, and $\Theta = P(X|Par(X))$ is a set of conditional probability distributions of the nodes $X \in V$ given its parental nodes denoted by $Par(X)$. Figure 1.2 illustrates how to use a DBN to model a simple three-nodes regulatory network (left) given in Fig. 1.1. In the time-invariant DBN model, the network structure is assumed stationary, and the state of each node measured at time $t + 1$ is dependent on the states of its parental nodes and itself measured at time t only. For example in Fig. 1.2, the node X_2's value at time t_2 is dependent on its parental nodes X_1 and X_3's values and itself at t_1.

The directed time-invariant DBN can be encoded by a joint distribution [31] over all the random variables $V = (\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_n})$:

$$\begin{aligned} P(V) &= P(\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_n}) = \textstyle\prod_{\mathbf{X} \in V} P(\mathbf{X}|Par(\mathbf{X})) \\ &= P(\mathbf{X_1})P(\mathbf{X_2}|\mathbf{X_1}) \times \ldots \times P(\mathbf{X_n}|\mathbf{X_{n-1}}), \end{aligned}$$

where, $P(\mathbf{X_t}|\mathbf{X_{t-1}}) = P(X_{t1}|Par(X_{t1})) \times \cdots \times P(X_{tp}|Par(X_{tp}))$, $Par(X_{tj})$ represents the gene j's parents' level measured at time $t - 1$.

Given the data $\mathbf{D}$, a scoring function is needed to evaluate the goodness of the network. Different scoring metrics, including the Bayesian Dirichlet equivalence (BDe) metric [28], BIC/AIC [2], Chow-Liu tree learning algorithm [8], have been applied to learn the structure of the dynamic Bayesian network. Bayesian Dirichlet equivalence (BDe) metric [28] is one of the most widely used methods to learn the network structures. Learning an optimal directed network (G^*, Θ) is equivalent to

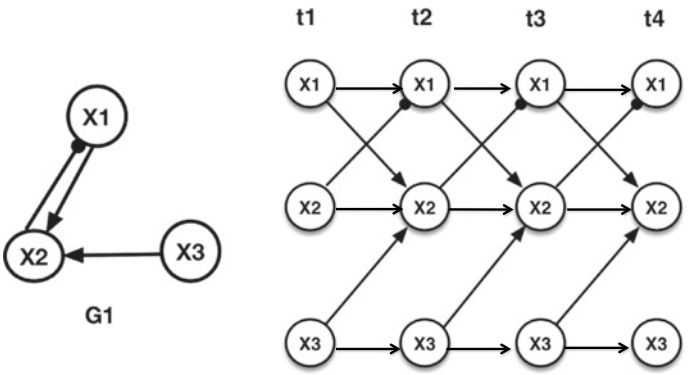

Fig. 1.2 Illustration of a three-nodes regulatory network (left) and a stationary dynamic Bayesian network model (right) whose structure is invariant with time. The state of each node measured at time $t + 1$ is dependent on the states of its parental nodes and itself measured at time t only

maximizing the posterior distribution of the network G, that is to solve the following optimization problem:

$$G^* = \text{argmax}_G\, P(G|\mathbf{D}) = \text{argmax}_G\, P(G, \mathbf{D}) = \text{argmax}_G\, P(\mathbf{D}|G)P(G),$$
$$P(\mathbf{D}|G) = \int P(\mathbf{D}|G, \Theta)P(\Theta|G)d\Theta,$$
$$\mathbf{D}|\Theta \sim Multinomial(\Theta),$$
$$\Theta|G \sim Dirichlet(\alpha).$$

where, $P(G)$ is the prior of the network G, which can be chosen in different ways, e.g, the minimal description length (MDL) was used in [17]. $P(\mathbf{D}|G)$ is the likelihood function if the parameter vector Θ is continuous. The structure parameters Θ's prior follows Dirichlet distribution with a hyperparameter vector α given a network G; while the $\mathbf{D}$ is a multinomial sample dependent on the parameters Θ; and Θ are assumed to be globally and locally independent. Later, some heuristic searching algorithms (e.g., greedy searching and simulated annealing which was used in the Banjo) are applied to find the optimal networks. Algorithm 2 summarizes the procedure to infer an optimal stationary dynamic Bayesian network model. It is noteworthy to mention that, the BDe metric has assumed the following two assumptions are valid: two directed acyclic networks G_1, G_2 are equivalent if they encode the same joint probability distribution; and, if the network G_1 is equivalent to G_2, the distribution function of Θ will be same in both networks.

Algorithm 2: BDe-based Stationary Network Structure Learning

Input: High-dimensional time-series data D
for *each network G* **do**
  Optimal network structure learning;
    $\mathbf{D}|\Theta \sim Multinomial(\Theta)$;
    Dirichlet prior distribution for $\Theta|G$;
    Estimate Bayesian Dirichlet equivalence (BDe) metric;
end
Optimal network searching:
  Sort networks according to BDe scores;
  Search optimal network by simulated annealing;
Output: Optimal regulatory networks

Another popular method is to maximize the BIC score which is written as $BIC(G) = logP(G|\mathbf{D}) - f(N)|G|$, where $f(N)$ is a non-negative penalization function, and $|G|$ denotes the network complexity. Thus, finding an optimal network requires trading off fit and complexity of network. It is also called minimal description length (MDL) score. If $f(N) = 1$, we get the Akaike Information

Criterion (AIC) scoring function [2]. Many network structure learning algorithms have been proposed in the past twenty years, the interested readers could refer to [2] and other references for details.

1.2.2.2 Time-Varying DBN Learning

Figure 1.3 illustrates time-varying regulatory networks and how to use the dynamic Bayesian networks to model these regulatory networks with different structures at different stages. The regulatory network experiences structure changes from stage G1 to G2 to G3, including the removal of edges (X1 activates X2), birth of new edges (X3 inhibits X1), and change of activation/inhibition relationship (X3 inhibits X2) at three different stages. If there is at least one change-point, the first-order Markov property will not be valid in the time-varying DBN because of the new network structure. Our recent work [43] proposed that, in the time-varying DBN model, the first-order Markov property should be updated as: the state of any node measured at time $t + 1$ is dependent on the states of itself and its parental nodes measured

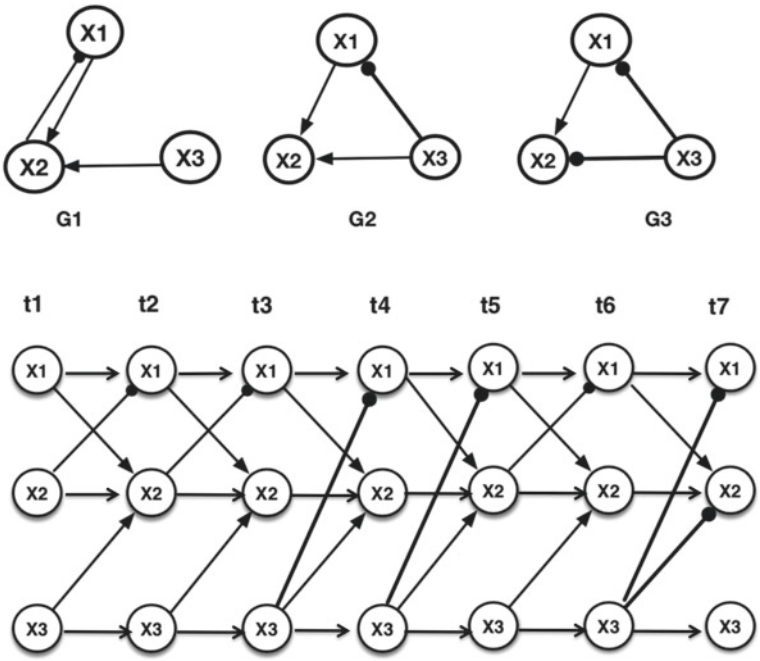

Fig. 1.3 Illustration of time-varying regulatory network (up), and time-varying dynamic Bayesian network model (bottom). The regulatory network experiences structure changes from stage G1 to G2 to G3, including the removal of edges (X1 activates X2), birth of new edges (X3 inhibits X1), and change of activation/inhibition relationship (X3 inhibits X2) at three different stages

at time t and also its current network structure or stage. So the structure parameter Θ which is described by the conditional probability distribution can be written as $\Theta = P(X|Par(X), S)$, where $X \in V$, and S represents the stage.

In the time-varying DBN learning, there will be different optimal network structures denoted by $G^* = (G_1^*, \ldots, G_{\nu+1}^*)$ at different stages $S = (S_1, \ldots, S_{\nu+1})$ if there exist ν change-points $(c_1, \ldots, c_\nu)$. To learn the optimal time-varying network structures is to maximize the joint posterior distribution of all the network structures which are stage dependent, the optimization problem is express as

$$\begin{aligned}(G_1, \ldots, G_{\nu+1})^* &= \text{argmax}_{(G_1, G_2, \ldots, G_{\nu+1})} P((G_1, \ldots, G_{\nu+1})|D, S) \\ &= \text{argmax}_{G=(G_1, G_2, \ldots, G_{\nu+1})} P(D|S)P(G|S)\end{aligned}$$

To directly solve the above optimization problem is very difficult, in our recent work [43], we first applied the INSPECT algorithm to identify the number and locations of change-points. That is, we know how many stages and when the structure starts to make changes. Then, we used the stationary DBN model to learn the network structures at different stages individually. That is, we assume these network structures are independent at different stages, so the above optimization problem can be simplified as

$$\begin{aligned}(G_1, \ldots, G_{\nu+1})^* &= \text{argmax}_{(G_1, G_2, \ldots, G_{\nu+1})} P((G_1, \ldots, G_{\nu+1})|(D_1, \ldots, D_{\nu+1}), (S_1, \ldots, S_{\nu+1})) \\ &= \text{argmax}_{G=(G_1, G_2, \ldots, G_{\nu+1})} P(G_1|D_1, S_1) \ldots P(G_{\nu+1}|D_{\nu+1}, S_{\nu+1})\end{aligned}$$

where, D_i represents the observation data at stage S_i. Algorithm 3 summarizes the pseudocode of the time-varying network structure learning.

Algorithm 3: Time-varying Network Structure Learning

Input: High-dimensional time-series data D;

1st, Identify Number and locations of change-points;
2nd, Divide the data D according to stages;

for *data* D_t *of the stage* S_t **do**
  for *each network* G_t **do**
    Stationary DBN network structure learning at stage S_t;
  end
  Optimal network searching;
end
Output: Optimal time-varying regulatory networks

Most existing network reconstruction methods based on the dynamic Bayesian network could only infer a causality or correlation graph, not a regulatory network. The regulatory network should contain the activation and inhibition information

to describe the regulatory relationship. Next, we will discuss how to identify the regulatory relationship on the inferred stationary and time-varying networks.

1.2.3 Regulatory Relationship Identification

Figures 1.2 and 1.3 illustrate time-invariant and time-varying regulatory networks. On the graph, an arrow is used to represent the activation event, while the filled ball arrow is used to represent the inhibition event. In our previous work [35], we introduced and estimated the signed integer weights, which modified the influence score proposed in [49], to identify the activation and inhibition relationship and interaction strength. If it is a time-invariant regulatory network, the identified activation/inhibition relationship on each edge will not change over time. A positive weight corresponds to an activation event, while a negative value corresponds to an inhibition event between two nodes. The Bayesian network inference with Java objects [49] estimated the influence score according to the cumulative distribution function which is written as

$$G_{ijk}(t) = \sum_{m=0}^{k} \omega_{ijm}(t) = \sum_{m=0}^{k} P(X_{ti} = m | Par(X_{ti}) = j). \tag{1.1}$$

$G_{ijk}(t)$ calculates the probability that gene X_{ti}'s level is no more than k given its parent gene takes a value of j. For the gene X_{ti} at time t, if ω_{ijm} is an increasing function, X_{ti}'s expression level has a high chance to be upgraded given that its parent's level increases. Then, the interaction between X_{ti} and its parent will be voted as an activation event by a predefined voting machine [49] based on the values of $G_{ijk}(t)$; else, it will be voted as an inhibition event. Our previous work [35] converted the influence score to be a signed integer weight, which can not only describe the regulatory relationship and interaction strength, but also update the state transfer function in the symbolic model checking.

In the time-varying networks, the structure changes could influence the sign and magnitude of interaction strength. Our recent work [43] extended the Eq. 1.1 to allow the integer weights to change at different stages in response to the network structure changes. Now the quantity $G_{ijk}(t, S_t)$ in Eq. 1.2 measures the probability that, at time t in the stage S_t, a node X_i will take a value $m \in \{0, 1, \ldots, k\}$ given that its parent nodes $Par(X_i)$ taking a value of j at stage S_t. If ω_{ijm} is an increasing (decreasing) function of its parent node's value, then, this interaction will be voted as an activation (inhibition) event at a specific stage S_t based on the values of $G_{ijk}(t, S_t)$. Finally, the influence score will be converted into a signed integer weights at different stages.

$$G_{ijk}(t, S_t) = \sum_{m=0}^{k} \omega_{ijm}(t, S_t) = \sum_{m=0}^{k} P(X_{ti} = m | Par(X_{ti}) = j, S_t). \tag{1.2}$$

The estimation of stage-dependent integer weight using the Eq. 1.2 is not easy compared with Eq. 1.1 if the change-points or stages are unknown. To simplify the problem, if we can identify the change-points in the first step using the change-points detection algorithm, we can estimate the integer weights of the networks at different stages individually using the Eq. 1.1. This simplification is based on the assumption that the network structures are independent at different stages. Algorithm 4 summarizes the procedure for the time-varying regulatory network structure learning and integer weight estimation.

Algorithm 4: Signed Integer Weight Estimation in Time-varying Networks

Input: High-dimensional time-series data D;

1st, Identify Number and locations of change-points;
2nd, Divide the data D according to stages;

for *data* D_t *of the stage* S_t **do**
  for *each network* G_t **do**
    Optimal stationary DBN network structure learning;
    Signed integer weight estimation;
    for *each edge* $(X_i, Par(X_i))$ **do**
      Compute $\omega_{ijm}(t, S_t) = P(X_{ti} = m | Par(X_{ti}) = j, S_t)$;
      Compute $G_{ijk}(t, S_t) = \sum_{m=0}^{k} \omega_{ijm}(t, S_t)$;
      Calculate Influence Score based on $G_{ijk}(t, S_t)$;
      Convert the influence score into signed integer weights;
    end
    Optimal network searching;
  end
end
Output 2: Optimal time-varying regulatory networks;
Signed integer weights at different stages.

Gene regulatory networks or signaling pathways models are normally complex and composed of thousands of genes or hundreds of proteins and interactions, but with a small number of observations, especially in the time-series data. All the network inference and change-points detection algorithms are dependent on some parameters. The inferred "optimal" network could be different if we change some parameters' values. After the networks are inferred, how to validate these "optimal" networks? Without validation, we can not trust the simulation results or predictions made by the inferred networks. Next section, we will introduce two major formal verification methods, symbolic model checker SMV and probabilistic model checker PRISM that have been applied in our previous studies, and discuss how to use them to formally verify the inferred networks.

1.3 Formal Analysis of Regulatory Networks

Model Checking [9] is a formal verification technique for the finite state systems modeled by a Kripke Structure, which is depicted by $M = (S, s_0, R, L)$, where S is a finite set of states with the initial state $s_0 \in S$, R is a transition relation between states, and L is a labeling function. During model checking, the model M will first be converted into a state transition system, then, model checker will automatically and exhaustively search the state transition system to verify or falsify the desired property, which is expressed as a temporal logic formula ψ, starting from the initial state s_0. Model Checking problem [9] is expressed as $\{s \in S | M, s \models \psi\}$, which means, the model M satisfies ψ. Different model checking tools, for example, BLAST, Prism, SPIN, NuSMV, et al., have been developed to verify the design of hardware and software systems. The model checking technique was also introduced to study the cyber-physical system [7, 38] and biological systems [19, 20, 23]. Our previous studies [19, 20, 22, 24, 25] have introduced and applied different model checking techniques, including statistical model checker, symbolic model checker SMV, and probabilistic model checker PRISM, to formally analyze signaling pathway models and gene regulatory networks. This technique has been very successful in hardware systems verification, according to our studies, it is very promising that it could be a powerful tool in biological network verification.

Though we had discussed the temporal logic formulas in our previous studies [19, 20, 22, 24, 25], we will revisit some key formulas and semantics in this section again for completeness.

1.3.1 Temporal Logic Formula

Temporal logic formulas ψ can be divided into two subtypes, Linear Temporal Logic (LTL) and Computation Tree Logic (CTL). The LTL or CTL formula ψ is composed of an atomic proposition AP, Boolean variables, Boolean connectives which include $\vee$ (or), $\wedge$ (and), $\neg$ (not), and $\rightarrow$ (implication), but in SMV or PRISM code, we use the following symbols | (or), & (and), ! (not) to encode the Boolean connectives. In the LTL formula, we also need the temporal operators **X, F, G, U** to describe some property on a path, in which, $\mathbf{X}p$ means p holds in the next state of the path; $\mathbf{F}p$—p holds at some state in the Future (eventually) on the path; $\mathbf{G}p$—p holds Globally (always) at every state on the path; $p\mathbf{U}q$—p holds Until q holds on the path. A CTL formula describes properties of computation trees [9], which have branches corresponding to all possible paths from the root (that is, initial state of the system). In the CTL formula, the path quantifier **A, E** must precede the LTL operators **X, F, G, U**, so there are eight CTL operators: AX, EX, AG, EG, AF, EF, AU, EU that are frequently used in our studies.

Given the state formulas ψ and path formulas ϕ, the syntax of the CTL logic is expressed as [9]

$$\psi ::= AP \mid \psi_1 \vee \psi_2 \mid \neg\psi \mid \mathbf{E}\phi \mid \mathbf{A}\phi$$
$$\phi ::= \mathbf{X}\psi \mid \mathbf{F}\psi \mid \mathbf{G}\psi \mid \psi_1 \mathbf{U} \psi_2.$$

The path $\pi = s_0, s_1, \ldots$ represents an infinite sequence of states, where $s_i \in S$, and for every $i \geq 0$, $(s_i, s_{i+1}) \in R$ represents a transition of the system, and π^i denotes the suffix of π starting at s_i. The semantics of a CTL formula is summarized below [9]

$M, s \models p$ iff $p \in L(s)$;
$M, s \models \neg\psi$ iff $M, s \models \psi$ does not hold;
$M, s \models \psi_1 \vee \psi_2$ iff $M, s \models \psi_1$ or $M, s \models \psi_2$;
$M, \pi \models \mathbf{X}\psi$ iff $M, \pi^1 \models \psi$;
$M, \pi \models \psi_1 \mathbf{U} \psi_2$ iff there exists $k \geq 0$ such that, $M, \pi^k \models \psi_2$ and for all $0 \leq j < k$, $M, \pi^j \models \psi_1$;
$M, s \models \mathbf{E}\phi$ iff there exists a path π from $s$such that $M, \pi \models \phi$;
$M, s \models \mathbf{A}\phi$ iff for every path π from s, $M, \pi \models \phi$.

The readers could refer to [9] for more details about the LTL and CTL semantics and formulas.

1.3.2 Symbolic Model Checking

The symbolic model verifier (SMV) encoded by the ordered binary decision diagram (OBDD) [6] is a powerful model checking technique that has been successfully used for the verification of CPU or digital circuits designs. Our previous work have applied SMV to verify different regulatory networks [35, 43] and signaling pathway models [20, 23]. Algorithm 5 describes the pseudocode of SMV program given a network and some experimental observations. Each program starts with a "MODULE MAIN", and all the variables (describing the genes/proteins) are defined by "VAR" and initialized by "init" under the keyword "ASSIGN". The state transition for any node for the next state is updated by a state transfer function which is dependent on the signed integer weights estimated by the Algorithm 4, the value of the current state and corresponding parental nodes' values.

After the network is encoded, finally, the keyword "SPEC" will be used to encode some desired CTL formulas which are abstracted from the experimental observations. SMV can exhaustively search the state transition system M and check whether $M \models \psi$ is true or not. If the model M satisfies the property ψ, SMV will output "True", else, "False" with a counter-example will be given.

The LTL and CTL formula discussed in the Sects. 1.3.1–1.3.2 can only describe properties of an infinite sequence of states. That is, there is no time bound. However, if M describes a stochastic model, in most simulation studies, we have to stop the simulation to check some properties, so there is a time bound. For example, we can use the software BioNetGen [25] to model the stochastic biochemical reactions, which

Algorithm 5: Symbolic Model Verification of Regulatory Networks

Part 1: SMV Code Structures

Input 1: Inferred networks M;
Signed integer weights;
Temporal logic formula ψ

for *each network,* **do**
MODULE MAIN: Starter of SMV code;
VAR: Declare variables;
ASSIGN & init: Initialize variables and assign values;
next: Update state by the transfer functions;
end

Output 1: SMV Code for each network

Part 2: SMV Verification of CTL Formula

Input 2 : SMV Code from Part 1;
Observations/Experimental results

1. Design desired CTL formula ψ from experiment;
2. **SPEC**: Specify the CTL formula ψ;
3. Attach CTL formula to the end of SMV code;
4. Run SMV Model Checker to verify $M \models \psi$.

Output 2: Network M satisfies ψ: True or False

will generate some traces or executions. The temporal logic formula describing these traces are time-bounded. Next, we will introduce the time-bounded linear temporal logic (BLTL).

1.3.3 Time-Bounded Linear Temporal Logic (BLTL)

Similar to the LTL formula, a time-bounded LTL (BLTL) formula is constructed by the atomic propositions (AP) using boolean connectives and bounded temporal operators. The syntax of the logic is expressed as

$$\phi ::= AP \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \neg\phi_1 \mid \phi_1 \mathbf{U}^t \phi_2.$$

There are three frequently used time-bounded operators: **F**, **G**, and **U**, which are defined as: $\mathbf{F}^t\phi$ or $\mathbf{F}(\leq t)[\phi]$ means ϕ holds true within time t; $\mathbf{G}^t\phi$ or $\mathbf{G}(\leq t)[\phi]$ means ϕ holds true globally up to time t; the time-bounded until operator $\phi_1 \mathbf{U}^t \phi_2$ or $\phi_1 \mathbf{U}(\leq t)\ \phi_2$ means, *within* time t, ϕ_1 will hold until ϕ_2 becomes true. The basic BLTL operators $\mathbf{F}^t$, $\mathbf{G}^t$, $\mathbf{U}^t$ can also be combined together to construct a composite operator to verify some complicated properties. In our work, we have used the formula $F^{t_1} G^{t_2}[\phi]$ to describe the property that ϕ holds true within time t_1 and will be globally

true up to time t_2. The semantics of time-bounded LTL is defined with respect to the trace of a stochastic system. If σ^k denotes the trace starting at the step k, then, $\sigma^k \models \phi$ means that, the track σ^k satisfies the bounded LTL formula ϕ. The semantics of time-bounded LTL is written as

- $\sigma^k \models AP$ if and only if AP holds true in s_k;
- $\sigma^k \models \phi_1 \vee \phi_2$ if and only if $\sigma^k \models \phi_1$ or $\sigma^k \models \phi_2$;
- $\sigma^k \models \phi_1 \wedge \phi_2$ if and only if $\sigma^k \models \phi_1$ and $\sigma^k \models \phi_2$;
- $\sigma^k \models \neg\phi_1$ if and only if $\sigma^k \models \phi_1$ does not hold;
- $\sigma^k \models \phi_1 \mathbf{U}^t \phi_2$ if and only if there exists $i \in N$ such that, for each $0 \leq j < i$, $\sigma^{k+j} \models \phi_1$, and if $\sum_{0 \leq l < i} t_{k+l} \leq t$, then $\sigma^{k+i} \models \phi_2$.

The interested readers could refer to [21, 25] for more details about the BLTL semantics and formulas.

1.3.4 Probabilistic Model Checker PRISM

Most cellular signaling pathways or genetic network stochastic simulation models are continuous-time stochastic processes. PRISM is a popular probabilistic model checker, which can automatically and formally model and verify three types of probabilistic models, including the discrete-time Markov chains, Markov decision processes, and continuous-time Markov chains (CTMCs) models, and the PRISM model file is given an extension .sm.

Algorithm 6: PRISM Model Checking Pseudocode

Part1 : ctmc // Continuous-Time Markov Chains Model: model.sm

Input: Probabilistic M;
Bounded temporal logic formula ψ
const double c = 0.1; // Declare constant c
const int N = 100; // Declare constant N

module NAME //Starter of a module NAME;
Gene1: [0..N] init N; // Initialize variable and assign values;
[] predicate → rates: updates // State update form;
[] geneA >0 → c*geneA: geneA' = geneA - 1 // Update geneA's state with a rate c*geneA
endmodule

Part 2: PRISM Verification: property.csl

const double T = 80; define a constant time T = 80 seconds;
const double p = 0.9; define a constant probability p = 90%;
P>=p [F <=T (geneA >= 50)]; check the property is true/false;
P=? [F<=T (geneA >= 50)]; estimate the probability;

Output: True/False, or Estimate the probability that ψ is true

Algorithm 6 summarizes the procedure to write PRISM code to verify a CTMC stochastic model. In our stochastic simulation models of cellular networks, the PRISM code starts with the keyword **ctmc**. We first should define some constants using the keyword **const**, for example, the reaction rates. All the PRISM programs should contain at least one module using the keyword "**module ... endmodule**", which contains all the variables and updates. The variables should be declared and initialized first. For example, we can define the possibles for the geneA using the code: "geneA: [0..N] **init** N;", which means, geneA takes an integer value ranging from 0 to N, and its initial value is specified (as N) with the keyword **init**.

PRISM can implement both synchronous and asynchronous update of modules, so it can model most biochemical reactions occurring asynchronously in the cell. The state update of each variable in the module are decided by the "predicate" and "updates" taking the form: [] **predicate** $\rightarrow$ **rates** : **updates**. That is, for each variable X, if the $predicate$ is true, then, the states of the corresponding variable X in the module will be updated "asynchronously" (using an "empty" square bracket []). The variable X's value will be updated to a new value X' according to the "$updates$" rules in the form of $X' = f(X)$ with a rate of "$rates$". The "$rates$" should be proportional to some known parameters (e.g., reaction rates) and the number of molecules that each variable represents.

In the continuous-time Markov chain models, we will use the continuous stochastic logic (CSL), which is a property specification language for CTMC process, to specify the temporal logic properties. The PRISM can formally verify two types of properties. The first type of property is to verify or falsify the formula $P_{\geq p}\ [\phi]$, which is an assertion, the answer could be true or false. PRISM applies the Wald's sequential probability ratio test (SPRT) to check a given CSL formula on-the-fly when the simulation traces can give an answer ("True", "False") at a given confidence level p. For example in the Algorithm 6, the property "$P_{>=p}\ [\mathbf{F}_{<=t}\ (geneA \geq 50)]$" means, within t seconds, the probability that "the number of geneA molecules in the cell will be no less than 50" is at least p. This verification procedure is similar to the hypothesis testing-based statistical model checking method [25, 47, 48]. The second type of CSL formula that PRISM can analyze is to use the confidence interval (CI) method to estimate a numerical value of a bounded property $P_{=?}[\phi]$, that is to estimate the probability that the formula could be true. For example, "$P_{=?}\ [\mathbf{F}_{<=t}\ (geneA \geq 50)]$" estimates the probability that the number of geneA molecules will be no less than 50 within t seconds. PRISM could run the simulation to check the property on the fly and estimate the probability.

Now, we have introduced change-points detection algorithm, stationary and time-varying dynamic Bayesian network structure learning algorithm, signed integer weights estimation algorithm, symbolic model checking technique SMV and probabilistic model verification technique PRISM. Next, we will discuss how to integrate all these methods in a unified framework to reconstruct regulatory networks.

1.4 Integrative Data Analysis

The Drosophila microarray data [4], which measures 4028 genes' expression levels of Drosophila at four stages: embryonic, larval, pupal periods, and adulthood with 67 time points, has been used to study the regulatory networks that are involved in the muscle development. Some algorithms [33, 40, 41] have been developed to reconstruct undirected networks associated with the muscle development. Since the number of time points is significantly smaller than the number of genes, some dimensional reduction methods have been used to select a small number of genes to study. In this section, we will discuss the procedure how to integrate these methods together to analyze the Drosophila data.

We have known the real change-points of the Drosophila microarray data [4] are at the location/index (31, 41, 59). In Fig. 1.4, we applied the stationary DBN to reconstruct a time-invariant network which is composed of 11 genes from all the 67 time-point observations. Some studies have indicated that the network structure should be different in the Drosophila's life cycle. Below, we will reconstruct the time-varying networks at different stages for comparison.

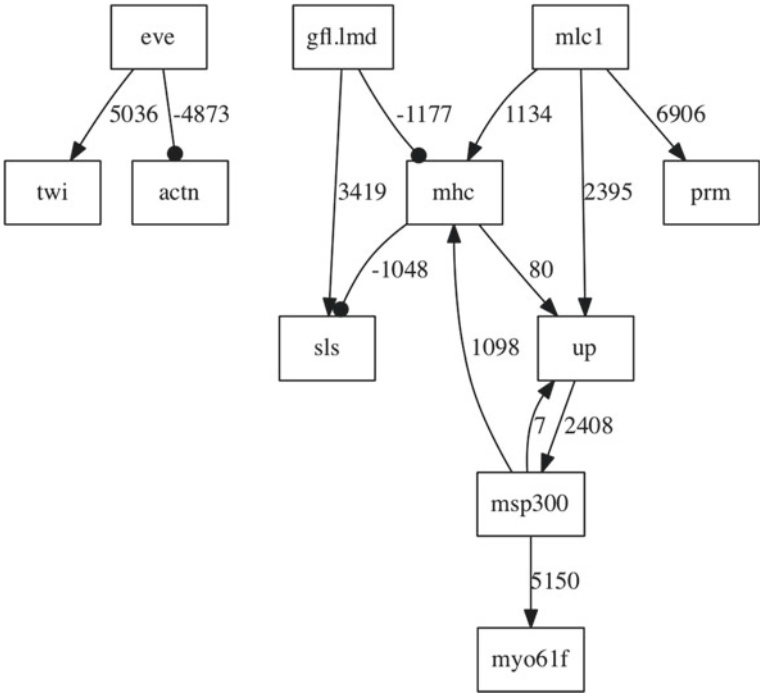

Fig. 1.4 Reconstruction of an optimal regulatory network using stationary DBN from all the 67 time-point observations in the Drosophila's life cycle

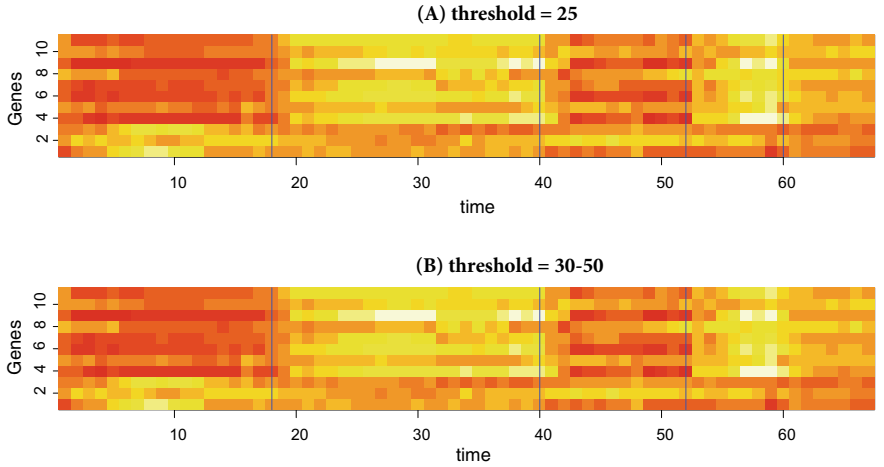

Fig. 1.5 Estimation of change-points from 11 genes, where the blue vertical lines represent the change-point lines with different threshold values (**A**: threshold = 25; **B**: threshold = [30, 50])

We will apply the INSPECT algorithm to infer the change-points assuming both the number and location of change-points are unknown. Most change-points detection algorithms are sensitive to some parameters when analyzing the high-dimensional data. INSPECT is sensitive to at least one parameter, for example, the parameter "threshold", which is used to test whether an identified change-point is a true changepoint.

We first identified the change-points from the low-dimensional dataset which has only eleven genes that are involved in the wing muscle development identified in the previous studies [12, 50]. Figure 1.5 plots the inferred change-points on the heatmap of 11 genes, the blue vertical lines represent the change-points. Figure 1.5A has 4 change-points (18, 40, 52, 60) with a threshold value 25, while Fig. 1.5B has 3 change-points (18, 40, 52) with a threshold value ranging within [30,50]. Our results indicate that the parameter's value influences the number of change-points only, it does not change the locations. Then, we test the performance of the INSPECT algorithm on high-dimensional data. We randomly sampled 400 genes from the 4028 genes and repeat 100 times and count the frequencies for the identified change-points. Figure 1.6 plots some histograms of estimated change-point locations using different threshold values. The results are similar to the Fig. 1.5, a smaller threshold value will lead to more change-points, while a larger threshold value will infer fewer change-points. To find a threshold value that is universal to any size of high-dimensional data is not realistic, our studies found when you change the data size, the previous threshold value will not work anymore.

After the change-points are identified, the next step is to apply dynamic Bayesian network inference method and signed integer weight estimation algorithm to infer optimal time-varying regulatory networks of Drosophila's muscle development.

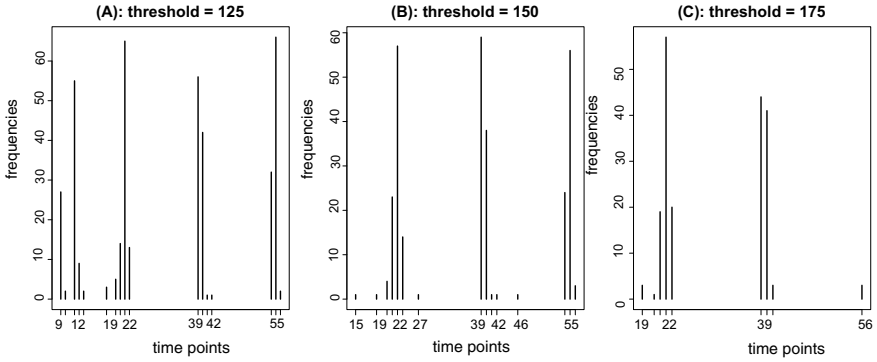

Fig. 1.6 Histograms of estimated change-point locations using different threshold values. A smaller threshold value will lead to more change-points (**A**), while a larger threshold value will infer fewer change-points (**C**)

Since the change-points are already known, the data can be splitted into four different subsets for the network reconstruction at different stages individually.

Figure 1.7 illustrates four optimal regulatory networks during the Drosophila's life cycle from the embryonic (A), larval (B), pupal (C) to adulthood (D). The solid lines with arrows represent activation, while the circle-head arrows represent inhibition. The integers on the directed edges are signed integer weights, which describe the interaction strength between two nodes and regulatory relationship. Apparently, these four optimal regulatory networks undergo systematic rewiring, that is, they are not invariant in the Drosophila's life cycle, but most regulatory relationships are still conserved. Figure 1.7 shows that, in the embryonic stage, the gene msp300 is an upstream gene, it can continuously activate several downstream genes, including the mlc1, up, eve, and myo61f to promote the embryonic development. But when the cell enters the larval, pupal, and adulthood stage, msp300's activities will be regulated by some of its previous downstream genes, for example, it could be inhibited by the mlc1 and but activated by the mhc gene. This result could explain the previous experimental discovery [45] that msp300 regulates the actin-dependent nuclear anchorage. Our previous work [43] first identified msp300 as a hub gene in the muscle development and explained the experimental discoveries.

This network reconstruction method is very sensitive to some parameters due to the small number of measurements, and this method could infer more than one optimal network at the same stage using the same data. Figure 1.8 illustrates another inferred four top-scoring non-identical networks at different stages. Comparison of Figs. 1.7 and 1.8 shows several differences in these optimal network structures. Our technique could output any number of top-scoring optimal regulatory networks. These networks are all "statistically" optimal, it does not mean that they are biologically correct. The next step is to verify or falsify the inferred network and choose the best network that is consistent with the experiments and used for further data analysis and simulation. Next, we will discuss how to apply SMV and PRISM model checker for the formal

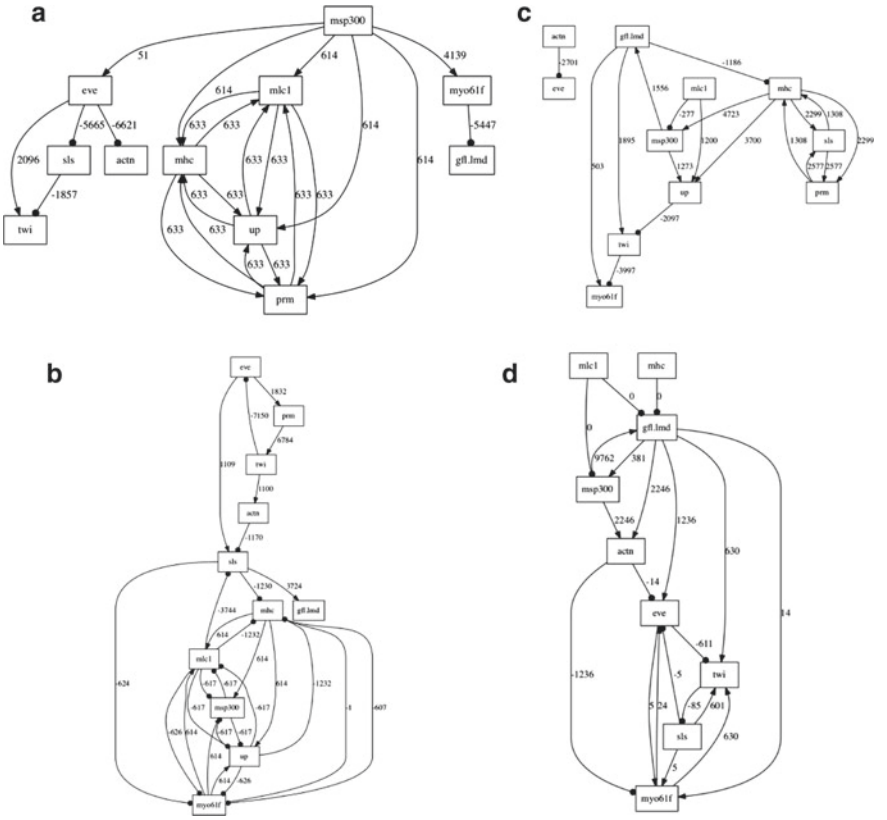

Fig. 1.7 Optimal regulatory networks during the Drosophila's life cycle from the embryonic (**A**), larval (**B**), pupal (**C**) to adulthood (**D**)

analysis of inferred networks. The below examples are used for demonstration only, the interested readers could refer to our previous work [21, 22, 35, 43] for more details.

Before the application of model checkers for network verification, we need to build a model first to describe the system. During the SMV network verification, we prefer to build a discrete value model with fewer parameters than other types of models, that is, each variable or gene can only take discrete values. For example, we can assume that each gene or variable can take three possible values $\{-1, 0, 1\}$, which represent down-regulated, normal, and up-regulated, and the initial state is randomly assigned a value of either 0 or -1. Note, we can also assume that the variable can take n possible values $\{1, 2, \ldots, n\}$ if needed. Then, we use state transfer functions to update the state of each variable at different stages. For example, the SMV code for the state update of the gene "twi" in the adulthood stage is not only dependent

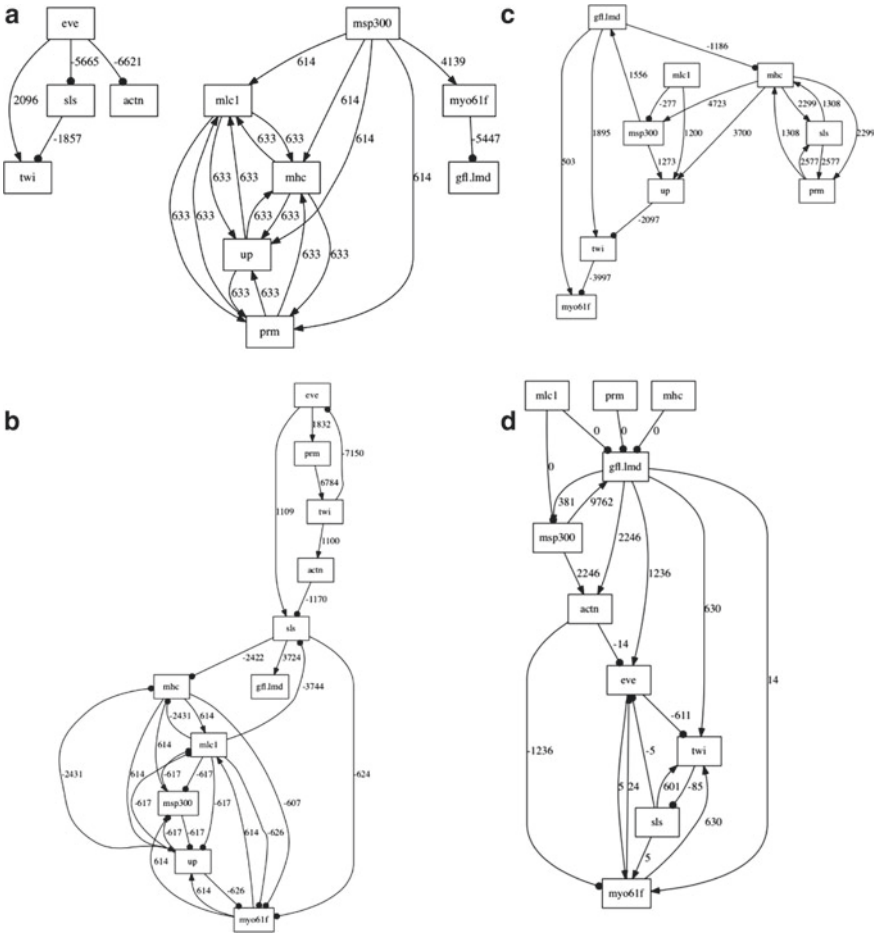

Fig. 1.8 Another four top-scoring regulatory networks during the Drosophila's life cycle

on the values of the parental genes myo61f, sls, glf.lmd, and eve, but also dependent on the signed integer weights:

```
next(twi) :=
          case
               630*myo61f + 601*sls + 630*gfl.lmd - 611*eve > 0  : 1;
               630*myo61f + 601*sls + 630*gfl.lmd - 611*eve = 0  : 0;
               630*myo61f + 601*sls + 630*gfl.lmd - 611*eve < 0  :-1;
          esac;
```

For illustration, we designed some putative CTL formulas to show how to translate some experimental results or existing database into temporal logic formulas for the SMV model checker to verify or falsify the inferred networks in Table 1.1.

Table 1.1 Putative CTL formulas for the regulatory networks verification

Property	CTL formula
P1	AG(msp300 > 0 → AX(mlc1 = 1 & up = 1 & eve = 1))
P2	AG(msp300 > 0 → AF(twi < 0 & sls < 0 & prm < 0))
P3	AG(msp300 > 0 → AF(twi > 0 & sls > 0 & prm > 0))
P4	EG((mlc1 = 1 | msp300 = 1 | gfl.lmd = 1) → EF(twi ≤ 0 & up ≥ 0))
P5	AG((mlc1 = 1 | msp300 = 1 | gfl.lmd = 1) → AF(twi ≤ 0 & up ≥ 0))
P6	AG(mhc=1 → AF(prm = 1 & mlc1 = 1 & up = 1))
P7	AG((mhc = 1 → msp300 = 1) → EF(msp300 = 1 & mhc ≤ 0))
P8	AG((mhc = 1 → AF(prm = 1)) & (prm = 1 → AF(sls = 1)) & (sls = 1 → AF(mhc ≥ 0)))
P9	AG((sls = 1) → AF(mhc = 1 & mhc ≤ 0))
P10	AG((msp300 = 1 → AF(mlc1 = -1)) & (mlc1 = 1 → AF(myo61f = -1)) & (myo61f = 1 → AF(msp = 1)))
P11	AG((sls = 1 → AF(myo61f = 1)) & (myo61f = 1 → AF(eve = -1)) & (eve = 1 → AF(twi = - 1)) & (twi = 1 → AF(sls = - 1)))

Our recent work [43] has verified or falsified several temporal logic formulas related to the time-varying regulatory network of Drosophila. We list some similar CTL formulas in Table 1.1 to explain how to construct temporal logic formulas and their meanings in biology, but we will not provide the verification results in this chapter. The interested readers could refer to [43] for similar examples and computer code. The formulas P1-P3 describe the properties related to the gene msp300. These formulas have the operators **AX** and **AF**. P1 means, the activation of msp300 will activate the genes mlc1, up, and eve if they are msp300's downstream genes. This type of formula could intelligently identify many downstream genes at the same time in the large network. P2-P3 are used to check whether or not there exists a path on which msp300's overexpression will finally inhibit (P2) or activate (P3) the genes twi, sls, and prm's expression levels. Similar formulas like the P1-P3 can be designed to check some hub genes and their downstream genes' behaviors. Formulas P4-P5 describe the properties of (twi, up)'s parental genes mlc1, msp300 and gfl.lmd, for some (P4) or all (P5) paths, it is globally true that either mlc1 or msp300 or gfl.lmd's overexpression will finally inhibit twi's activity but promote the gene up's expression. The formula P4 with the "EF" operator is weaker than P5 with the "AF" operator in verification. P6 checks whether or not the mhc's overexpression will finally promote the genes prm, mlc, and up's activity. Property 7 describes a negative feedback loop between mhc and msp300. P8-P11 describes a sequence of reaction events, which can be easily expressed using the CTL formula, then verified or falsified by the SMV model checker. These properties are not easy to be simulated using traditional computational methods (e.g., stochastic simulation or differential equations). Using the traditional simulation methods, you have to estimate many unknown parameters if the computational models are complicated which is very challenging.

Our SMV verification technique can avoid using some unknown parameters. In our previous work, we have applied this technique to identify some key biomarkers and processes that can initiate the cell's apoptosis or proliferation. If we can translate the KEGG database into CTL formulas, the SMV model checker could automatically verify/falsify whether or not the inferred networks are consistent with the KEGG database, instead of manually checking with the database. If the formula is satisfied, SMV will output "True", else it will output "False" with a counter-example. The networks satisfying all or most properties will be biologically correct or reasonable, which can be used for further analysis or modeling and predictions.

Finally, we will obtain one or more than one inferred optimal networks that are verified to be consistent with most existing experiments by the SMV model checker. Sometimes, given some small networks, we can use stochastic continuous models to describe the inferred optimal networks. The probabilistic model checker PRISM, based on sequential probability ratio test and confidence interval estimation methods, can formally analyze some quantitative properties of the system described by the bounded temporal logic formulas.

Table 1.2 designed some putative BLTL formulas for the PRISM analysis if we could build a stochastic continuous model. Formula P1 means that, within time t, the number of "prm" molecules will be at least 100 with a probability of at least 0.9. Formula P2 checks whether the number of "msp300" molecules will be continuously greater than 50 from time 10 to t. PRISM can check whether these formulas are true or not with different values of time t. PRISM can also estimate the probability that the number of "prm" molecules will be at least 100 within time t, which is expressed as P3. P4 will estimate the probability that the msp300's level will be above 50 all the time during the time interval [10, 30]. However, the continuous models contain many unknown parameters than the discrete value models, including the rates of reaction, synthesis, binding or degradation. Most of these parameters are not easy to be estimated from the experiments. The formulas in Table 1.2 are used for demonstration only since we do not know the parameters in the inferred regulatory network. We usually use PRISM to formally analyze some small networks with known parameters. Our previous work [21] has applied PRISM to formally analyze a stochastic signaling pathway model, the interested readers could refer to [21] for computer code and more details.

Table 1.2 Two types of putative BLTL formulas used for the PRISM model verification. These formulas are used for demonstration only

	BLTL formula
Assertion P1	$P_{\geq 0.9}(\phi_1) = P_{\geq 0.9}\ [F_{\leq t}\ (prm \geq 100)]$
Assertion P2	$P_{\geq 0.9}(\phi_2) = P_{\geq 0.9}\ \{G_{[10,t]}\ (msp300 > 50)\}$
Estimation P3	$P_{=?}(\phi_3) = P_{=?}\ [F_{\leq t}(prm \geq 100)]$
Estimation P4	$P_{=?}(\phi_4) = P_{=?}\ \{G_{[10,30]}\ (msp300 > 50)\}$

1.5 Discussions

In this work, we discussed how to integrate the machine learning and model checking methods to reconstruct regulatory network from the high-dimensional data. In the integrative approach, the change-points detection algorithm is first applied to identify the change-points which describe when the system moves to a new stage; then, the dataset is splitted into subsets for individual analysis according to the stages. The stationary dynamic Bayesian network method is applied to learn the optimal network structure at different stages; in the meantime, a signed integer weight estimation algorithm is used to learn the regulatory relationship (activation/inhibition). There could be several inferred optimal networks using the current network inference methods, then SMV model checker will be applied to formally verify the inferred time-varying networks by checking some temporal logic formulas which are abstracted from experiment or KEGG. The optimal and biologically correct network should be consistent with existing experiments and known databases. Given a continuous stochastic model, PRISM was introduced for further analysis of the networks by checking some time-bounded temporal logic formulas. However, PRISM is only applicable to some small networks with all parameters already known to us because it is not realistic to estimate many model parameters in large networks.

This work and our recent study [43] found that the change-points detection algorithm is sensitive to the parameter's values and the data size in the high-dimensional time-series data analysis. It is not possible to find a universal parameter value for the change-points estimation according to our analysis. So, the change-points detection is still one of the most challenging problems in network reconstruction. During the time-varying network structure learning, we assume the network structures are independent at different stages, so we can split the data according to the change-points and infer the networks and estimate the integer weights at different stages individually. However, the networks might not be independent at different stages. In the long run, we need to develop a real time-varying network structure learning algorithm that does not assume the stage independence, and propose novel change-points estimation algorithms to overcome the shortcomings of previous methods in the time-varying network reconstruction.

Acknowledgements This work was partially supported by the NIH-NIGMS grant 1R15GM129696-01A1.

References

1. Ahmed A, Xing E (2009) Recovering time varying networks of dependencies in social and biological studies. Proc Natl Acad Sci 106:11878–11883
2. Akaike H (1974) A new look at the statistical model identification. IEEE Trans Autom Control 19
3. Akutsu T, Miyano S, Kuhara S (2000) Inferring qualitative relations in genetic networks and metabolic pathways. Bioinformatics 16:727–734

4. Arbeitman M, Furlong E, Imam F, Johnson E et al (2002) Gene expression during the life cycle of drosophila melanogaster. Science 297:2270–5
5. Barry D, Hartigan J (1992) Product partition models for change point problems. Ann Stat 20(1):260–279
6. Bryant R (1986) Graph-based algorithms for boolean function manipulation. IEEE Tran Comput 35(8):677–691
7. Ceccarelli M, Cerulo L, Santone A (2014) De novo reconstruction of gene regulatory networks from time series data, an approach based on formal methods. Methods 69(3):298–305
8. Chow C, Liu C (1968) Approximating discrete probability distributions with dependence trees. IEEE Trans Info Theory 14
9. Clarke EM, Grumberg O, Peled DA (1999) Model checking. MIT Press
10. Darling D, Erdos P (1956) A limit theorem for the maximum of normalized sums of independent random variables. Duke Math J 23:143–155
11. Doering T et al (2012) Network analysis reveals centrally connected genes and pathways involved in cd8+ t cell exhaustion versus memory. Immunity 37
12. Dondelinger F, Lebre S, Husmeier D (2013) Non-homogeneous dynamic bayesian networks with bayesian regularization for inferring gene regulatory networks with gradually time-varying structure. Mach Learn 90
13. Fearnhead P (2006) Exact and efficient Bayesian inference for multiple change point problems. Stat Comput 16(2):203–213
14. Feuz K, Cook D, Rosasco C, Robertson K, Schmitter-Edgecombe M (2014) Automated detection of activity transitions for prompting. IEEE Trans Human-Mach Syst 45(5):1–11
15. Friedman J, Hastie T, Tibshirani R (2007) Sparse inverse covariance estimation with the graphical lasso. Biostatistics, pp 1–10
16. Friedman N, Linial M, Nachman I, Pe'er D (2000) Using bn to analyze expression data. J Comput Biol 7:601–620
17. Friedman N, Murphy K, Russell S (1998) Learning the structure of dynamic probabilistic networks. In: Proceedings of the 14th conference on the uncertainty in artificial intelligence
18. Fujita A, Sato J et al (2007) Time varying modeling of gene expression regulatory networks using the wavelet dynamic vector autoregressive method. Bioinformatics 23(13):1623–1630
19. Gong H (2013) Analysis of intercellular signal transduction in the tumor microenvironment. BMC Syst Biol 7:S5
20. Gong H, Feng L (2014) Computational analysis of the roles of er-golgi network in the cell cycle. BMC Syst Biol 8:S4
21. Gong H, Feng L (2014) Probabilistic verification of er stress-induced signaling pathways. In: Proceedings of IEEE international conference on bioinformatics and biomedicine
22. Gong H, Klinger J, Damazyn K, Li X, Huang S (2015) A novel procedure for statistical inference and verification of gene regulatory subnetwork. BMC Bioinform V16:S7
23. Gong H, Wang Q, Zuliani P, Clarke E (2011) Formal analysis for logical models of pancreatic cancer. In: 50th IEEE conference on decision and control and European control conference
24. Gong H, Zuliani P, Komuravelli A, Faeder J, Clarke E (2012) Computational modeling and verification of signaling pathways in cancer. Proceedings of algebraic and numeric biology, LNCS 6479
25. Gong H, Zuliani P, Komuravelli A, Faeder JR, Clarke EM (2010) Analysis and verification of the HMGB1 signaling pathway. BMC Bioinform 11(7)
26. Green P (1995) Reversible jump markov chain monte carlo computation and bayesian model determination. Biometrika 82:711–732
27. Grzegorczyk M, Husmeier D (2009) Nonstationary continuous dynamic bayesian networks. Adv Neural Inf Process Syst (NIPS) 22:682–690
28. Heckerman D, Geiger D, Chickering D (1995) Learning bayesian networks: the combination of knowledge and statistical data. Mach Learn 20(3)
29. Horwitz B (2003) The elusive concept of brain connectivity. NeuroImage 19:466–470
30. Kawahara Y, Sugiyama M (2009) Sequential change-point detection based on direct density-ratio estimation. In: SIAM international conference on data mining, pp 389–400

31. Kim S, Imoto S, Miyano S (2003) Inferring gene networks from time series microarray data using dynamic bayesian networks. Brief Bioinform 4:228–235
32. Kim S, Imoto S, Miyano S (2004) Dynamic bayesian network and nonparametric regression for nonlinear modeling of gene networks from time series gene expression data. BioSystems 75:57–65
33. Lebre S, Becq J, Devaux F, Stumpf M, Lelandais G (2010) Statistical inference of the time-varying structure of gene-regulation networks. BMC Syst Biol 4:130
34. Luscombe N et al (2004) Genomic analysis of regulatory network dynamics reveals large topological changes. Nature 431:308–312
35. Ma Y, Damazyn K, Klinger J, Gong H (2015) Inference and verification of probabilistic graphical models from high-dimensional data. Lect Notes Bioinform 9162
36. Mazumder R, Hastie T (2012) The graphical lasso: new insights and alternatives. Electron J Stat 6:2125
37. Ong I, Glasner J, Page D (2002) Modelling regulatoruypathways in e. coli from time series expression profiles. Bioinformatics 18:S241–S248
38. Parvu O, Gilbert D (2016) A novel method to verify multilevel computational models of biological systems using multiscale spatio-temporal meta model checking. PLOS One
39. Reddy S, Fun M, Burke J, Estrin D, Hansen M, Srivastava M (2010) Using mobile phones to determine transportation modes. ACM Trans Sens Netw 6(2):1–27
40. Robinson J, Hartemink R (2010) Learning non-stationary dynamic bayesian networks. J Mach Learn Res 11:3647–3680
41. Schwaller L, Robin S (2016) Exact bayesian inference for off-line change-point detection in tree-structured graphical models. Stat Comput 27(5)
42. Wang T, Samworth R (2017) High dimensional change point estimation via sparse projection. In: Statistical Methodology
43. Wang Z, Guo Y, Gong H (2019) An integrative analysis of time-varying regulatory networks from high-dimensional data. In: IEEE international conference on big data, pp 3798–3807
44. Yoshida R, Imoto S, Higuchi T (2005) Estimating time-dependent gene networks form time series microarray data by dynamic linear models with markov switching. CSB05, IEEE CSBC
45. You J, Starr D, Wu X, Parkhurst S, Zhuang Y, Xu T, Xu R, Han M (2006) The kash domain protein msp-300 plays an essential role in nuclear anchoring during drosophila oogenesis. Deve Biol 289(2):336–45
46. You Y, Wang T, Samworth R (2015) A useful variant of the davis-kahan theorem for statisticians. Biometrika 102:315–323
47. Younes HLS, Simmons RG (2002) Probabilistic verification of discrete event systems using acceptance sampling. CAV, LNCS 2404:223–235
48. Younes HLS, Simmons RG (2006) Statistical probabilistic model checking with a focus on time-bounded properties. Inf Comput 204(9):1368–1409
49. Yu J, Smith V, Wang P, Hartemink A, Jarvis E (2004) Advances to bayesian network inference for generating causal networks from observational biological data. Bioinformatics 20:3594–3603
50. Zhao W, Serpedin E, Dougherty E (2006) Inferring gene regulatory networks from time series data using the minimum description length principle. Bioinformatics 22