Vol.2015-MPS-103 No.65
Vol.2015-BIO-42 No.65
2015/6/25

# Exact and Heuristic Methods for Network Completion for Time-Varying Genetic Networks

Natsu Nakajima[1,a)] Tatsuya Akutsu[1,b)]

***Abstract:*** A biological system maintains its functions against internal and external perturbations, leading to topological changes in the network with varying delays. To understand the flexibility of biological networks, we propose a novel approach to analyze time-dependent networks, based on the framework of network completion. In this report, we present a novel completion method for time-varying networks based on a double dynamic programming technique to detect change time points and required modifications. Although this method allows us to guarantee the optimality of the solution, it has low computational efficiency. In order to resolve this difficulty, we developed a heuristic method for reducing the computation time of minimum least-squared errors. We demonstrate the effectiveness of our proposed methods through computational experiments using synthetic data and microarray gene expression data.

## 1. Introduction

Gene regulatory networks play important roles in cells and maintain organism functions through protein production, response to the external environment, and control of cell division processes. Therefore, deciphering gene regulatory network structures is important for understanding cellular systems, which might also be useful for the prediction of adverse effects of new drugs and the detection of target genes for the development of new drugs. Many existing studies have focused on the use of gene expression profiles and these network models mostly assume that the topology of the network does not change through time, whereas the actual gene regulatory network might dynamically change its structure depending on time, the effects of certain shocks, etc. Recently, many reverse engineering tools have been proposed, which can reconstruct time-varying biological networks based on time-series gene expression data. Yoshida et al. [1] developed a dynamic linear model with Markov switching that represents change points in regimes that evolve according to a first-order Markov process. Fujita et al. [2] proposed a method based on the dynamic autoregressive model. This model extends the vector autoregression (VAR) model, which can be applied to the inference of non-linear time-dependent biological correlations such as dynamic gene regulatory networks. Robinson and Hartemink [3] proposed a model called a non-stationary dynamic Bayesian network, based on dynamic Bayesian networks, which allows inference from data generated by non-stationary processes in a time-dependent manner. Lèbrel et al. [4] also introduced the autoregressive time-varying (ARTIVA) algorithm for the analysis of time-varying network topologies from time-course data generated from different processes. This model adopts a combination of reversible jump Markov chain Monte Carlo (RJMCMC) and dynamic Bayesian network (DBN), in which RJMCMC is used for the detection of change time points and the resulting networks, and DBN is used to represent causal interactions among genes. Thorne and Stumpf [5] presented a method to model the regulatory network structure between distinct segments with a set of hidden states by applying the hierarchical Dirichlet process hidden Markov model [6], including a potentially infinite number of states and a Bayesian network model for estimating relationships between genes. Rasool and Bouaynaya [7] presented a new method based on constrained and smoothed Kalman filtering, which is capable of estimating time-varying networks from time-series data, including unobserved and noisy measurements. The dynamics of genetic modules are represented as a linear-state space equation and the observability of linear time-varying systems is defined by imposing sparse constraints in Kalman filters. Ahmed et al. [8] proposed an algorithm called Tesla with machine learning, which can be cast in the form of a convex optimization problem. The basic assumption in this method is that networks at close time points do not have significant topological differences but have common edges with high probability, in contrast, networks at distant time points are markedly different. The regulatory networks are represented by Markov random fields at arbitrary time intervals.

In our recent study, we proposed a new method DPLSQ, for the analysis of time-independent networks, called network completion [9], [10], in which the minimum amount of modifications are made to a given network so that the resulting network is most consistent with the observed data.

This report presents two novel methods for the completion and inference of time-varying networks, DPLSQ-TV and DPLSQ-HS. DPLSQ-TV is an extension of DPLSQ [10] such that it

[1] Bioinformatics Center, Institute for Chemical Research, Kyoto University Gokasho, Uji, Kyoto 611-0011, Japan
[a)] nakajima@kuicr.kyoto-u.ac.jp
[b)] takutsu@kuicr.kyoto-u.ac.jp

can detect the change time points at which the network structure changes. Since the additions and deletions of edges are basic modifications in network completion, we need to extend DPLSQ so that these operations can be performed at several time points. In DPLSQ-TV, required modifications and change points are detected by a novel double dynamic programming algorithm in which the outer loop is used to detect change points and the inner loop is used to identify static network structures between the corresponding time intervals. Although DPLSQ-TV provides an optimal solution in polynomial time if the maximum indegree is bounded by a constant, the degree of the polynomial is not low, which prevents the method from being applied to the completion of large-scale networks. Therefore, we further propose a heuristic method DPLSQ-HS, to reduce the computation time by imposing a limit on the number of combinations of incoming nodes.

We evaluate the performance of our methods through computational experiments using synthetic data and microarray gene expression data from the life-cycle of *D. melanogaster* and the cell-cycle of *S. cerevisiae.*

## 2. Method

We assume that there exist $m$ time points $(1, 2, \cdots, m)$, which are divided into $B+1$ intervals: $[1, \cdots, c_1 - 1]$, $[c_1, \cdots, c_2 - 1]$, $\cdots$, $[c_B, \cdots, m]$, where $B$ indicates the number of change points. A different network is associated with each interval. We assume that the set of genes does not change, therefore, only the edge set changes according to the time interval. Let $V = \{v_1, \cdots, v_n\}$ be the set of genes. Let $E$ be the initial set of directed edges (i.e., initial set of gene regulation relationships), and $E_0, E_1, \cdots, E_B$ be the sets of directed edges (i.e., the output), where $E_i$ denotes the edge set for the $i$-th interval.

Then, the problem is defined as: given an initial network $G(V, E)$ consisting of $n$ genes, $N$ time-series datasets, each of which consists of $m$ time points for $n$ genes and the positive integers $h$, $w$, and $B$; infer $B$ change points (i.e., $c_1, c_2, \cdots, c_B$) and complete the initial network $G(V, E)$ by adding $h$ edges and deleting $w$ edges in total such that the total least-squared error is minimized. This results in the set of edges $E_0, E_1, \cdots, E_B$ at the corresponding time intervals (see **Fig. 2**). It is to be noted that if we start with an empty set of edges (i.e., $E = \emptyset$), the problem corresponds to the inference of a time-varying network.

### 2.1 Model of Differential Equation and Estimation of Parameters

We assume that the dynamics of each node $v_i$ are determined by the following differential equation:

$$\frac{dx_i}{dt} = a_0^i + \sum_{j=1}^{d} a_j^i x_{i_j} + \sum_{1 \le j < h \le d} a_{j,h}^i x_{i_j} x_{i_h} + b^i \omega, \qquad (1)$$

where $v_{i_1}, \cdots, v_{i_h}$ are incoming nodes to $v_i$, $x_i$ corresponds to the expression value of node $v_i$, and $\omega$ denotes random noise. The second and third terms of the right-hand side of the equation represent the linear and nonlinear effects to node $v_i$, respectively (see **Fig. 1**) [11].

In practice, we replace the derivative of Eq. (1) by the difference, and ignore the noise term as follows:

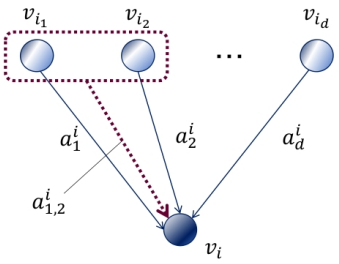


**Fig. 1** Dynamics model for a node. The expression level of $v_i$ is determined by the correlation between other genes (i.e., $v_{i_1}, \cdots, v_{i_d}$). $a_{1,2}^i$ is a coefficient corresponding to cooperative regulation by $v_{i_1}$ and $v_{i_2}$.

$$x_i(t+1) = x_i(t) + \left( a_0^i + \sum_{j=1}^{d} a_j^i x_{i_j}(t) + \sum_{1 \le j < h \le d} a_{j,h}^i x_{i_j}(t) x_{i_h}(t) \right). \qquad (2)$$

Then, the parameters $a_j^i$s and $a_{j,h}^i$s are estimated from these time-series data by minimizing the following objective function (i.e., the sum of least-squared errors) for each node $v_i$:

$$\sum_{t=1}^{m} \left| y_i(t+1) - \left[ y_i(t) + \left( a_0^i + \sum_{j=1}^{d} a_j^i y_{i_j}(t) + \sum_{1 \le j < h \le d} a_{j,h}^i y_{i_j}(t) y_{i_h}(t) \right) \right] \right|^2. \qquad (3)$$

### 2.2 Completion by Addition and Deletion of Edges

We present a new method for network completion of time-varying networks by the addition and deletion of edges [11].

Let $\sigma_{h_j, w_j, j}[p, q]$ denote the minimum sum-of-squared error for the time interval between $p$ and $q$ when adding $h_j$ edges to $e^-\left(v_j\right)$ and deleting $w_j$ edges from $e^-\left(v_j\right)$ as below, where the added and deleted edges must be disjointed.

$$\sigma_{h_j, w_j, j}[p, q] = \min_{\substack{j_1, j_2, \ldots, j_{h_j} \\ j'_1, j'_2, \ldots, j'_{w_j}}} \left\{ S^j_{e^-(v_j) - \{v_{j'_1}, v_{j'_2}, \ldots, v_{j'_{w_j}}\} \cup \{v_{j_1}, v_{j_2}, \ldots, v_{j_{h_j}}\}}[p, q] \right\}, \qquad (4)$$

where $\{v_{j'_1}, v_{j'_2}, \ldots, v_{j'_{w_j}}\}$ is the set of deleted edges from $e^-(v_j)$. We constrain the maximum $h_j$ and $w_j$ to the small constants $H$ and $W$ and let $\sigma_{h_j, w_j, j}[p, q] = +\infty$ if $h_j > H$, $w_j > W$, $h_j - w_j + |e^-\left(v_j\right)| \ge n$, or $h_j - w_j + |e^-\left(v_j\right)| < 0$ hold. Then, the problem is stated as

$$\min_{\substack{c_1 < c_2 < \cdots < c_B \\ h^0 + h^1 + \cdots + h^B = h \\ w^0 + w^1 + \cdots + w^B = w}} \left\{ \sum_{i=0}^{B} \min_{\substack{h_1 + h_2 + \cdots + h_n = h^i \\ w_1 + w_2 + \cdots + w_n = w^i}} \left[ \sum_{j=1}^{n} \sigma_{h_j, w_j, j}[c_i, c_{i+1} - 1] \right] \right\}. \qquad (5)$$

Here, we define $D[h, w, i, p, q]$ as

$$D[h, w, i, p, q] = \min_{\substack{h_1 + h_2 + \cdots + h_i = h \\ w_1 + w_2 + \cdots + w_i = w}} \left\{ \sum_{j=1}^{i} \sigma_{h_j, w_j, j}[p, q] \right\}. \qquad (6)$$

The elements of $D[h, w, j, p, q]$ can be computed by

$$\begin{aligned} D[h, w, 1, p, q] &= \sigma_{h, w, 1}[p, q], \\ D[h, w, j+1, p, q] &= \min_{\substack{h' + h'' = h \\ w' + w'' = w}} \left\{ D[h', w', j, p, q] + \sigma_{h'', w'', j+1}[p, q] \right\}. \end{aligned} \qquad (7)$$

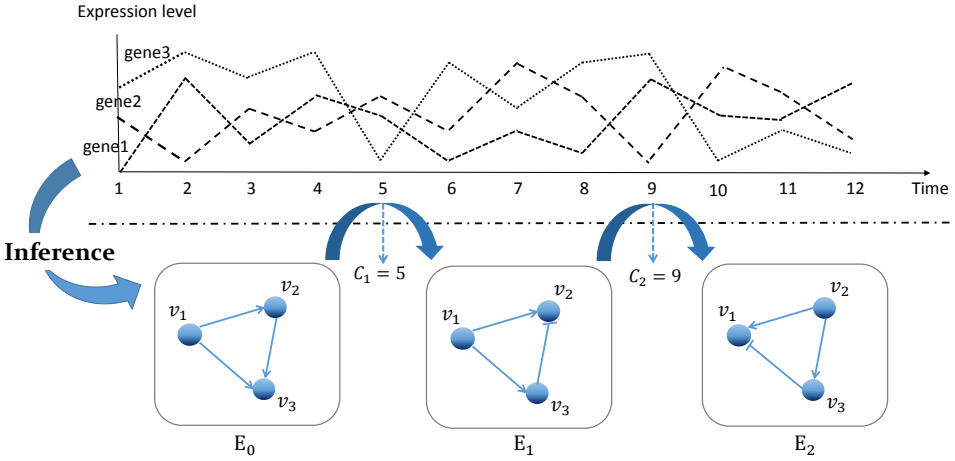


**Fig. 2** Overview of the network inference for time-varying genetic networks. Inference (i.e., completion starting with the null network) of time-varying structure during each time interval. This example corresponds to the case of $N = 1$, $n = 3$, $B = 2$, $m = 12$. The change points are $c_1 = 5$ and $c_2 = 9$.

Next, we define $E[h, w, b, q]$ as

$$E[h,w,b,q] = \min_{\substack{c_1<c_2<\cdots<c_b \\ h^0+h^1+\cdots+h^b=h \\ w^0+w^1+\cdots+w^b=w}} \left\{ \sum_{i=0}^{b} \min_{\substack{h_1+h_2+\cdots+h_n=h^i \\ w_1+w_2+\cdots+w_n=w^i}} \left[ \sum_{j=1}^{n} \sigma_{h_j,w_j,j}[c_i, c_{i+1}-1] \right] \right\}, \tag{8}$$

where $b$ is the number of change points and $c_{b+1} - 1 = q$. $E[h, w, b, q]$ can be computed by the following DP algorithm:

$$E[h,w,0,q] = D[h,w,n,1,q],$$
$$E[h,w,b,q] = \min_{\substack{p\in\{1,\cdots,q-1\} \\ h'+h''=h \\ w'+w''=w}} \{E[h',w',b-1,p] + D[h'',w'',n,p,q]\}. \tag{9}$$

## 3. Heuristic Method

Although the exact method, DPLSQ-TV, is guaranteed to find an optimal solution in polynomial time, the higher degree polynomial prevents it from being applied to the completion of large-scale networks. The reason why DPLSQ-TV requires a lot of CPU time is that least-squared errors are calculated for each node by considering all possible combinations of incoming nodes and taking the minimum value of them. Therefore, we propose a heuristic algorithm DPLSQ-HS, to improve the computational efficiency by relaxing the optimality condition. In order to improve the computational efficiency, we impose an upper limit on the number of combinations of incoming nodes. Although DPLSQ-HS does not guarantee an optimal solution, it allows us to reduce the computation time of computing of the minimum least-squares.

### 3.1 Schematic illustrations of DPLSQ-HS

Although DPLSQ-HS can be applied to completion by adding and deleting edges, we schematically show the procedure only for additions of edges, because we impose an upper limit only on the number of adding edges. In particular, we have developed DPLSQ-HS, which contributes to reducing the time complexity, by imposing the constraint on the number of combinations of incoming edges to each node. In **Fig. 3**, the diagram indicates that, for each node $v_i$, we maintain $M$ combinations of $h$ incoming nodes with $M$ lowest errors at the $h$-th step. Let $S_i^h$ denote the set of $M$ combinations computed at the $h$-th step for $v_i$. At the $h$-th step, for each combination $\{v_{i_1}, \ldots, v_{i_{h-1}}\} \in S_i^{h-1}$ where $i_1 < i_2 < \cdots < i_{h-1}$, we calculate the least-squared error for each $v_j$ such that $j > i_{h-1}$ is the $h$-th incoming node to $v_i$. The calculated least-squared errors are sorted in ascending order, the top $M$ values are selected, and the corresponding combinations are stored in $S_i^h$.

### 3.2 Algorithm

In the following, we give a detailed description of the algorithm to compute $\sigma_{h,i}[p, q]$ in DPLSQ-HS, where $\sigma_{h,i}[p, q]$ does not necessarily mean the minimum value, and the meaning of 'Step' is different from that in Section 3.1.

**Step 1:** For each period $[p, q]$, repeat Steps 2-7.
**Step 2:** Let $S_i^0 = \{\emptyset\}$ for all $i = 1, \ldots, n$.
**Step 3:** For $i = 1$ to $n$ do Steps 4-7.
**Step 4:** Repeat Steps 5-7 for each node $v_i$ from $h = 1$ to $h = H$.
**Step 5:** For each combination $\{v_{i_1}, \ldots, v_{i_{h-1}}\} \in S_i^{h-1}$ and each node $v_j$ such that $j > i_{h-1}$ ($j > 0$ if $h = 1$), calculate the least-squared error for the $h$ edge set $\{(v_{i_1}, v_i), \ldots, (v_{i_{h-1}}, v_i), (v_j, v_i)\}$.
**Step 6:** Sort the obtained least-squared errors in ascending order and select the top $M$ combinations, which are stored in $S_i^h$.
**Step 7:** Let $\sigma_{h,i}[p, q]$ be the minimum least-squared error among these top $M$ combinations.

The other parts of the algorithm are the same as in DPLSQ-TV.

## 4. Results and Discussion

We performed computational experiments using both synthetic data and microarray expression data. All experiments were performed on a PC with an Intel Core(TM)2 Quad CPU (3.0 GHz). We employed the liblsq library (http://www2.nict.go.jp/aeri/sts/stmg/K5/VSSP/install_lsq.html) for the least squares fitting method.

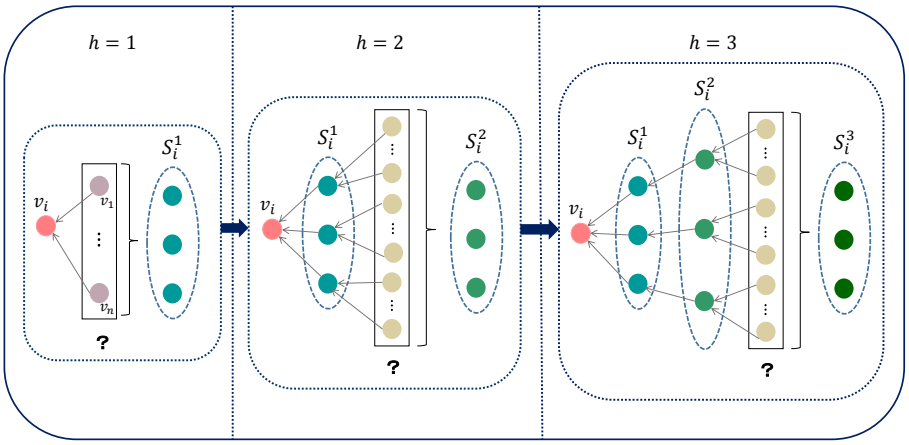


**Fig. 3** Schematic illustrations of the definition of the top $M$ combinations. This is an example that the case of $M = 3$ and $h \leq 3$. Let $S_i^h$ denote the set of $M$ combinations computed at the $h$-th step. At the $h$-th step, for each combination $\{v_{i_1}, \ldots, v_{i_{h-1}}\} \in S_i^{h-1}$, we calculate the least-squared error for each $v_j$ such that $j > i_{h-1}$ as a $h$-th incoming node to $v_i$.

### 4.1 Completion Using Synthetic Data

In order to evaluate the potential effectiveness of DPLSQ-TV and DPLSQ-HS, we start with network completion for time-varying networks using synthetic data. We test our performance in terms of detecting change time points that the sum-of-squared errors are minimized. We employed a randomly generated network consisting 10 genes as an initial network $G$, and three different networks $G_1$, $G_2$ and $G_3$ generated by randomly adding and deleting edges from the initial network. In this method, for each node $v_i$ with $d$ input nodes, we considered the following model:

$$x_i(t+1) = x_i(t) + \left( a_0^i + \sum_{j=1}^{d} a_j^i x_{i_j} + \sum_{1 \leq j < h \leq d} a_{j,h}^i x_{i_j}(t) x_{i_h}(t) + b_i \omega \right), \tag{10}$$

where $a_j^i$s and $a_{j,h}^i$s are constants selected uniformly at random from $[-0.5, 0.5]$ and $[-0.05, 0.05]$, respectively. For the details of the artificial generation of the observed data, refer to [11].

As for the time-series data, we generated an original dataset with 30 time points including two change points $c_1 = 10$, $c_2 = 20$, which is generated by merging three datasets for $G_1$, $G_2$ and $G_3$ [11]. We conducted computational experiments by DPLSQ-TV and DPLSQ-HS in which the initial network $G$ was modified by randomly adding $h_0$ edges and deleting $w_0$ edges per node, resulting in $G_1$, $G_2$ and $G_3$ using the default values of $h_0 \leq 2$, $w_0 \leq 2$ and $H = W = 2$. We evaluated the performance of two methods in terms of the accuracy of modified edges, the time point errors for time intervals, and the execution time for completion (CPU time). Furthermore, in order to examine how CPU time changes as the size of the network grows, we generated networks with 20, 30, 40 and 60 genes by making 2, 3, 4 and 6 copies of the original networks. We examined observation error levels of 0.05, 0.1, 0.3 and 0.5 for each of which we took the average time point errors, accuracies, and CPU time over 10 random modifications.

The accuracy is defined as follows:

$$\frac{h + w + \sum_{i=0}^{B} \left( \left| E_i \cap E_i' \right| - |E_i| \right)}{h + w}, \tag{11}$$

where $E_i$ and $E_i'$ $(i = 0, 1, \cdots, B)$ are the sets of edges in the original network and the completed network in each time interval, respectively. This value is 1 if all the added and deleted edges are correct and 0 if none of the added and deleted edges is correct [11]. The time point error means the average distance between the observed and estimated values for change time points and is defined as

$$\frac{1}{B} \sum_{i=1}^{B} \left| c_i - c_i' \right|, \tag{12}$$

where $c_i'$ $(i = 1, 2, \cdots, B)$ are estimated change points. As for the execution time, we show the average CPU time.

The results of the two methods are summarized in **Table 1**. As can be seen from this table, the time point errors are almost zero regardless of the network size and the level of observation errors. In addition, we are able to observe that the CPU time using DPLSQ-TV rapidly increases as the size of the network grows, but on the other hand, the CPU time by DPLSQ-HS increases gradually in case that the size of the network is more than 20. In particular, the DPLSQ-HS algorithm is nearly $4-5$ times and $8-10$ times faster than the DPLSQ-TV algorithm in case of 40 and 60 genes, while maintaining reasonably good accuracy. Hence, DPLSQ-TV and DPLSQ-HS can accurately detect change time points and that they can complete given networks by modifying the edges with relatively good accuracy if the error levels are not very large.

It must be noted that DPLSQ-HS worked reasonably fast even for $n = 60$ although DPLSQ-TV took about 40000 seconds per execution. However, the accuracies on DPLSQ-HS became around 0.3 even if the observation error level was low (i.e., $o^i = 0.05/0.1$). Therefore, DPLSQ-HS has limited applicability with respect to the accuracy of modified edges, although it may still be useful for networks with $n = 60$ if the main purpose is to detect change time points.

In addition, we carried out another experiments with varying parameters (one experiment per parameter) in order to examine the relationship between the number of change points $B$ and the maximum number of added and deleted edges for each

**Table 1** Result on completion with synthetic data

**(a)** Using DPLSQ-TV

| | | Observation error level | | | |
|---|---|---|---|---|---|
| | | 0.05 | 0.1 | 0.3 | 0.5 |
| $n = 10$ | Time point error | 0.00 | 0.00 | 0.00 | 0.00 |
| | Accuracy | 0.464 | 0.474 | 0.401 | 0.329 |
| | CPU time (sec.) | 102.211 | 121.340 | 135.762 | 119.680 |
| $n = 20$ | Time point error | 0.00 | 0.00 | 0.00 | 0.00 |
| | Accuracy | 0.347 | 0.332 | 0.326 | 0.324 |
| | CPU time (sec.) | 1721.682 | 1392.793 | 1427.215 | 1380.156 |
| $n = 30$ | Time point error | 0.00 | 0.00 | 0.00 | 0.00 |
| | Accuracy | 0.325 | 0.317 | 0.248 | 0.245 |
| | CPU time (sec.) | 5263.687 | 4389.434 | 4114.544 | 4174.860 |
| $n = 40$ | Time point error | 0.00 | 0.00 | 0.00 | 0.00 |
| | Accuracy | 0.366 | 0.315 | 0.246 | 0.253 |
| | CPU time (sec.) | 10993.607 | 10658.169 | 10631.024 | 10702.756 |
| $n = 60$ | Time point error | 0.00 | 0.00 | 0.00 | 0.00 |
| | Accuracy | 0.330 | 0.329 | 0.287 | 0.280 |
| | CPU time (sec.) | 59413.264 | 44507.814 | 36793.397 | 35799.802 |

**(b)** Using DPLSQ-HS

| | | Observation error level | | | |
|---|---|---|---|---|---|
| | | 0.05 | 0.1 | 0.3 | 0.5 |
| $n = 10$ | Time point error | 0.00 | 0.00 | 0.00 | 0.00 |
| | Accuracy | 0.363 | 0.464 | 0.393 | 0.308 |
| | CPU time (sec.) | 71.399 | 86.104 | 94.550 | 84.207 |
| $n = 20$ | Time point error | 0.00 | 0.00 | 0.00 | 0.00 |
| | Accuracy | 0.338 | 0.286 | 0.289 | 0.275 |
| | CPU time (sec.) | 463.798 | 456.222 | 478.047 | 474.031 |
| $n = 30$ | Time point error | 0.00 | 0.00 | 0.00 | 0.00 |
| | Accuracy | 0.343 | 0.279 | 0.211 | 0.239 |
| | CPU time (sec.) | 795.251 | 1277.968 | 1323.173 | 1308.003 |
| $n = 40$ | Time point error | 0.00 | 0.00 | 0.00 | 0.00 |
| | Accuracy | 0.342 | 0.278 | 0.224 | 0.272 |
| | CPU time (sec.) | 1865.118 | 2149.137 | 2224.621 | 2315.528 |
| $n = 60$ | Time point error | 0.00 | 0.00 | 0.00 | 0.00 |
| | Accuracy | 0.341 | 0.334 | 0.249 | 0.330 |
| | CPU time (sec.) | 3947.643 | 4397.677 | 4547.498 | 4384.758 |

node, $H$ and $W$ on the least-squared errors. The resulting sum-of-squared errors (i.e., $E[\ldots]$s) for DPLSQ-TV are 3.105, 3.321, 3.518, 2.451, and 2.651 for $(B, H, W) = (2, 1, 1), (3, 1, 1), (4, 1, 1), (2, 2, 2)$, and $(2, 4, 2)$, respectively. From this result, the use of larger $H$, $W$ resulted in smaller least-squared errors. It is reasonable that more parameters resulted in better least-squares fitting. However, use of larger $B$ did not result in smaller least-squared errors. It may be because addition of unnecessary change points increases the error if an enough number of edges are not added. It must be noted that although the least-squared errors are reduced, use of larger $H$, $W$ is not always appropriate because it needs much longer CPU time and may cause overfitting.

Finally, we also compared our results with those obtained by the ARTIVA algorithm [4], which is an accessible tool for the inference of time-varying networks. We applied ARTIVA to the synthetic datasets that were generated in the same way as for our proposed methods. The results show that ARTIVA can only detect one change point among three change points regardless of the level of the observation error [11], where ARTIVA does not uniquely detect change points but output probabilities of change points.

### 4.2 Inference Using Real Data

We tested the performance of DPLSQ-TV and DPLSQ-HS using two types of microarray data measured during the cell-cycle of *S. cerevisiae* and the life-cycle of *D. melanogaster*, and also compared our results with those obtained using the ARTIVA algorithm.

The first microarray dataset is the expression time-series collected by Spellman et al. [12]. We selected 10 genes associated with part of the cell-cycle network extracted from the KEGG database [13]. As for time-series data, we combined and employed four sets of time-series data (alpha, cdc15, cdc28, and elu) in [12] that were obtained in four different experiments with 71 time points including three change time points.

The second microarray dataset is the time-series taken from Arbeitman et al. [14], which contains the expression levels of 4028 genes. We selected 30 genes (TFs) comprised of TFs cascade [15] and used the datasets with 67 successive time points spanning four distinct stages: embryonic (31 time points), larval (10 time points), pupal (18 time points), and adulthood (8 time points) in the *D. melanogaster* life-cycle, which includes three change points [11]. In this experiment, we used the default values $H = 3$, $W = 0$ and $H = 2$, $W = 0$ for the cell-cycle data and the life-cycle data, respectively.

**Table 2** Result on inference of change points in *S. cerevisiae* data

| | $c_i$ (Correct answer) | $c'_i$ (DPLSQ-TV) | $c'_i$ (DPLSQ-HS) | ARTIVA |
|---|---|---|---|---|
| $i = 1$ | 25 | 25 | 25 | 24 |
| $i = 2$ | 40 | 40 | 40 | – |
| $i = 3$ | 56 | 56 | 56 | 60 |
| CPU time (sec.) | | 12359.57 | 2689.49 | – |

**Table 3** Result on inference of change points in *D. melanogaster* data

| | $c_i$ (Correct answer) | $c'_i$ (DPLSQ-TV) | $c'_i$ (DPLSQ-HS) |
|---|---|---|---|
| $i = 1$ | 31 | 19 | 19 |
| $i = 2$ | 41 | 31 | 31 |
| $i = 3$ | 60 | 42 | 42 |
| CPU time (sec.) | | 59789.33 | 11039.48 |

Since the actual time-varying networks still remain unclear, we only evaluated the time point errors and the average CPU time.

**Table 2** and **Table 3** show the results for the time point errors and the CPU time, where $c_i$s are values of change point in the original data and $c'_i$s are estimated values. As can be seen from Table 2, there seems to be no difference between the results of DPLSQ-TV and DPLSQ-HS, which can correctly detect the change points where the network topology changes. Moreover, the CPU time required for DPLSQ-HS is about 4 times faster than that needed for DPLSQ-TV. The analysis of *D. melanogaster* shows that both methods can reasonably detect the change points as listed in Table 3. At first glance, readers may think that the errors are large at all change point positions. However, both methods could almost exactly detect two change points, excluding the case of $i = 3$. From the point of view of computational time, DPLSQ-HS appreciably outperforms DPLSQ-TV, it runs about 5 times faster than DPLSQ-TV. Therefore, DPLSQ-HS allows us to decrease the computation time considerably. These results suggest that, in many cases, DPLSQ-HS can be expected to provide nearly-optimal solution at least for change time points and a considerable reduction in computational time.

Next, in comparative analysis with ARTIVA, we employed

**Table 4** Comparative experiment for the inference of change points

| | $c_i$ (Correct answer) | $c'_i$ (DPLSQ-TV) | $c'_i$ (DPLSQ-HS) | ARTIVA |
|---|---|---|---|---|
| $i = 1$ | – | 19 | 19 | 18 - 19 |
| $i = 2$ | 31 | 31 | 31 | 31 - 33 |
| $i = 3$ | 41 | 42 | 42 | 41 - 43 |
| $i = 4$ | 60 | 56 | 56 | 59 - 61 |
| CPU time (sec.) | | 1213444.45 | 26988.79 | – |

both microarray datasets and attempted to identify the change time points. The results from the yeast microarray data are shown in Table 2. Unlike our inference, ARTIVA could detect nearly precisely two change points, 24 and 60, among three change points, but the second could not. Lèbrel et al. [4] also examined the number of identified change points with *D. melanogaster* data. According to this validation, it has been reported that the time intervals 18 – 19, 31 – 33, 41 – 43 and 59 – 61 contain more than 40% of all change points. The results are summarized in **Table 4**. ARTIVA appears to have slightly better with respect to the inference of change points than our proposed methods. However, ARTIVA does not detect change time positions but determines time intervals at which the network topology might change. Therefore, DPLSQ-HS is more suited for the detection of change time positions at all time points.

### 4.3 Discussions and Conclusion

We addressed the problem of network completion for time-varying genetic networks from time-series data and proposed two novel methods for solving this problem. Based on the idea of DPLSQ, we developed DPLSQ-TV such that it can perform the modification operations at several time points. In order to detect the change time points and sets of modified edges in network completion, we developed two different types of double DP algorithms. The first algorithm DPLSQ-TV, is intended for exact completion and inference of time-varying networks. Although DPLSQ-TV allows us to guarantee the optimality of its solution, it suffers from computational inefficiency as the size of the network grows. In order to improve the computational efficiency of DPLSQ-TV, we developed an heuristic method DPLSQ-HS, to reduce the time complexity of calculating least-squared errors by imposing a limit on the number of combinations of incoming nodes. We showed that each of these two methods works in polynomial time if the maximum indegree is bounded by a constant.

The results of computational experiments reveal that the two methods can detect change points almost precisely and can infer the modified edges with reasonable accuracy. DPLSQ-TV can be expected to provide wide range of applications not only in network completion but also in network inference. Additionally, DPLSQ-HS allowed us to give a relatively good performance in terms of change point detection with reduced time complexity. These results indicate that, in many cases, DPLSQ-HS also enables us to provide near-optimal solutions without increasing the time complexity.

The issue to be tackled is to take into account the relationship between $G_i$ and $G_{i+1}$. Although $G_i$ and $G_{i+1}$ are inferred independently from the original network $G$ by the proposed methods, there should be some strong relationship between them. Therefore, such an extension is also important future work.

**Acknowledgments** The authors would like to thank Prof. Hideo Matsuda in Osaka University and Takanori Hasegawa in Tohoku University for helpful discussions. This work was partially supported by JSPS, Japan (Grants in-Aid 22240009 and 22650045).

### References

[1] Yoshida, R., Imoto, S., and Higuchi, T.: Estimating time-dependent gene networks from time series microarray data by dynamic linear models with Markov switching, *Proceedings of Computational Systems Biology*, pp. 289-298, 2005.

[2] Fujita, A., Sato, J. R., Garay-Malpartida, H. M., Morettin, P. A., Sogayar, M.C., Ferreira, C. E.: Time-varying modeling of gene expression regulatory networks using the wavelet dynamic vector autoregressive method, *Bioinformatics*, vol. 23, no. 13, pp. 1623-1630, 2007.

[3] Robinson, J., and Hartemink, A.: Non-stationary dynamic Bayesian networks, *Neural Information Processing Systems*, pp. 1369-1376, 2008.

[4] Lèbrel, S., Becq, J., Devaux, F., Stumpf, M. P. H., and Lelandais, G.: Statistical inference of the time-varying structure of gene-regulation networks, *BMC Systems Biology*, vol. 4, no. 130, 2010.

[5] Thorne, T., and Stumpf, M. P. H.: Inference of temporally varying Bayesian Networks, *Bioinformatics*, vol. 28, pp. 3298-3305, 2012.

[6] Teh, Y. W., and Jordan, M. I.: Hierarchical Bayesian nonparametric models with applications, *In Bayesian Nonparametrics. Cambridge University Press, Cambridge*, pp. 158-207, 2010.

[7] Rassol, G., and Bouaynaya, N.: Inference of time-varying gene networks using constrained and smoothed Kalman filtering, *International Workshop on Genomic Signal Processing and Statistics*, pp. 172-175, 2012.

[8] Ahmed, A., Song, L., and Xing, E. P.: Time-varying networks: recovering temporally rewiring genetic networks during the lofe cycle of drosophila, *SCS Technical Report Collection*, CMU-ML-08-118, 2008.

[9] Akutsu, T., Tamura, T., and Horimoto, K.: Completing networks using observed data, *Proceedings of 20th International Conference on Algorithmic Learning Theory*, pp. 126-140, 2009.

[10] Nakajima, N., Tamura, T., Yamanishi, Y., Hiromoto, K., and Akutsu, T.: Network completion using dynamic programming and least-squares fitting, *The Scientific World Journal*, vol. 2012, 957620, 2012.

[11] Nakajima, N., and Akutsu, T.: Exact and heuristic methods for network completion for time-varying genetic networks, *BioMed Research International*, vol. 2014, Article ID 684014, 14 pages, 2014.

[12] Spellman, P. T., Sherlock, G., Zhang, M. Q., Iyer, V. R., Anders, K., Eisen, M. B., Brown, P. O., Botstein, D., and Futcher. B.: Comprehensive identification of cell-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization, *Molecular Biology of the Cell*, vol. 9, no. 12, pp. 3273-3297, 1998.

[13] Kanehisa, M., Goto, S., Furumichi, M., Tanabe, M., and Hirakawa, M.: KEGG for representation and analysis of molecular networks involving diseases and drugs, *Nucleic Acids Research*, vol. 38, no. 1, pp. D355-D360, 2009.

[14] Arbeitman, M. N., Furlong, E. E., Imam, F., Johnson, E., Null, B. H., Baker, B. S., Krasnow, M. A., Scott, M. P., Davis, R. W., and White, K. P.: Gene expression during the life cycle of Drosophila melanogaster, *Science*, vol. 297, no. 5590, pp. 2270-2275, 2002.

[15] Song, L., Kolar, M., and Xing, E. P.: KELLER: estimating time-varying interactions between genes, *Bioinformatics*, vol. 25, no. 12, pp. i128-i136, 2009.