Journal of Quality Technology
A Quarterly Journal of Methods, Applications and Related Topics

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/ujqt20

Multi-node system modeling and monitoring with extended directed graphical models

Dengyu Li & Kaibo Wang

To cite this article: Dengyu Li & Kaibo Wang (2024) Multi-node system modeling and monitoring with extended directed graphical models, Journal of Quality Technology, 56:1, 38-55, DOI: 10.1080/00224065.2023.2229458

To link to this article: https://doi.org/10.1080/00224065.2023.2229458

Published online: 19 Jul 2023.

JOURNAL OF QUALITY TECHNOLOGY
2024, VOL. 56, NO. 1, 38–55
https://doi.org/10.1080/00224065.2023.2229458

Multi-node system modeling and monitoring with extended directed graphical models

Dengyu Li[a] and Kaibo Wang[a,b]

[a]Department of Industrial Engineering, Tsinghua University, Beijing, China; [b]Vanke School of Public Health, Tsinghua University, Beijing, China

ABSTRACT

Complex manufacturing systems usually contain a large number of variables. Dominated by certain engineering mechanisms, these variables show complicated relationships that cannot be effectively expressed by simple correlation matrices or functions, thus increasing the difficulty of modeling and monitoring these systems. The directed graphical model (DGM) has been used as a flexible tool for describing the relationship among variables in complex systems. However, the DGM treats all variables equally and fails to consider the structural information among them that usually exists. To address this problem, an extended directed graphical model (EDGM) and related parameter estimation, monitoring, and structure learning methods are proposed in this work. Taking prior engineering knowledge into consideration, the EDGM assigns variables into groups and uses groups of variables as nodes in the graph model. By adding hidden state variables to each node, the EDGM can effectively represent the relationship within and between nodes and provide promising monitoring performance. Numerical experiments and a real-world case study of the monocrystalline silicon growth process are performed to verify the effectiveness of the proposed methods.

KEYWORDS

directed graphical models; EWMA control charts; graph structure learning; penalized loss function; quality prediction

1. Introduction

Due to the quick development of design, manufacturing, and sensing techniques, a complex system usually contains a large number of components, which are represented by multiple variables in an industrial big-data environment. These types of variables are essential for characterizing system status, reflecting quality performance, and expressing process changes. Dominated by certain engineering mechanisms, these variables usually have complicated relationships, and it is fundamentally important to identify the complex relationship among the variables to facilitate quality monitoring and improvement practice. There are two major challenges in modeling these kinds of complex systems: (1) the number of variables in a system can be large; (2) the variation propagation mechanism and relationships among the variables are complicated. A proper model for capturing the relationship in a complex system is critical in quality engineering, and thus, has drawn much attention in the literature.

Stream of variation (SoV) has been one important way to model the variation propagation mechanism in multistage manufacturing systems (MMSs) (Shi 2006). In an MMS, variables from the same stage are parallel, while variables from different stages have a sequential structure; a state-space model is used to connect multiple stages together and explicitly show how process variables, stage states, and quality outputs affect each other. A sequential manufacturing process composed of N stages, which are numbered in ascending order is considered. A two-level linear state-space model is given to model the process quality (Huang and Shi 2004) as follows:

$$\begin{aligned} y_{k,i} &= a_k x_{k,i} + \epsilon_{k,j} \\ x_{k,i} &= b_k x_{k-1,i} + \sigma_{k,j} \end{aligned} \quad [1]$$

where $k = 1, \ldots, N$ represents the stages, $i = 1, 2, \ldots$ represents the product index, and $y_{k,i}$ and $x_{k,i}$ represent the observable quality measurement and unobservable quality characteristic, respectively. The coefficients a_k and c_k are usually estimated by certain engineering knowledge (Ding, Ceglarek, and Shi 2002). By introducing the state-space method, the SoV model effectively models variation propagation in sequential or parallel-sequential structures.

When engineering knowledge and the physical law of the processes are not perfectly known, some data-

CONTACT Kaibo Wang kbwang@tsinghua.edu.cn Department of Industrial Engineering, Tsinghua University, Beijing 100084, China.

driven methods, such as regression models (Deng and Jin 2015; Sun et al. 2016; Yan et al. 2021) and neural network models (Bukkapatnam et al. 2006), have also proven to be effective. For example, for MMSs, data-driven methods estimate quality outputs by using information from all variables from upstreaming stages as follows:

$$y_{k,i} = f_k\left(\{x_{j,i}\}_{j=1}^{k}, \ \{y_{j,i}\}_{j=1}^{k-1}; \boldsymbol{\theta}_k\right) \quad [2]$$

where $y_{k,i}$ and $x_{k,i}$ have similar meanings to Eq. [1] and $\boldsymbol{\theta}_k$ is the parameter set that needs to be estimated from observable data. Specific forms of $f_k(\cdot)$ separate the data-driven methods into different models, where linear regression models take linear forms of f_ks while neural networks use hidden layers to model complicated relationships between inputs and outputs. With more accurate modeling methods, monitoring techniques also have greater power detecting out-of-control situations based on the estimated relationships between variables.

Compared to the aforementioned SoV and data-driven methods, a graphical model can provide a flexible and suitable expression of the relationship between essential quality outputs (Gómez, Paynabar, and Pacella 2021; Sun, Huang, and Jin 2017) from systems with complex structures. To illustrate the probabilistic representation of complex systems, directed graphical models (DGMs) have proven to be an effective and commonly used method (Wang et al. 2021). A directed graphical model usually contains a set of nodes $\mathcal{N}$, and a set of directed edges, that is, an arc set, $\mathcal{A}$. An arc from one node to another indicates the existence of conditional dependence behind these two nodes. By assuming arc function forms, DGMs are able to describe a complex joint distribution of variables in systems using multivariate conditional distribution with specific graph structures. More details about DGMs can be found in previous work (de Campos 1998; Heckerman, Geiger, and Chickering 1994).

Although they provide a flexible expression of systems with complex structures, existing graphical models are limited in that one node represents only one physically observable variable, and all variables are equally and separately treated, with each one acting as one node in the graph. However, it is easily understood that the variables' behavior is driven by the engineering mechanism behind them. A system consists of multiple subsystems, and each subsystem is an integration of multiple components. Naturally, variables in a complex system can form different groups; the importance of the variables is different to the system, and the mechanisms that dominate the variables in the same group and variables in different groups can be different. Therefore, a model that can take this information into consideration is important for effective process modeling and monitoring.

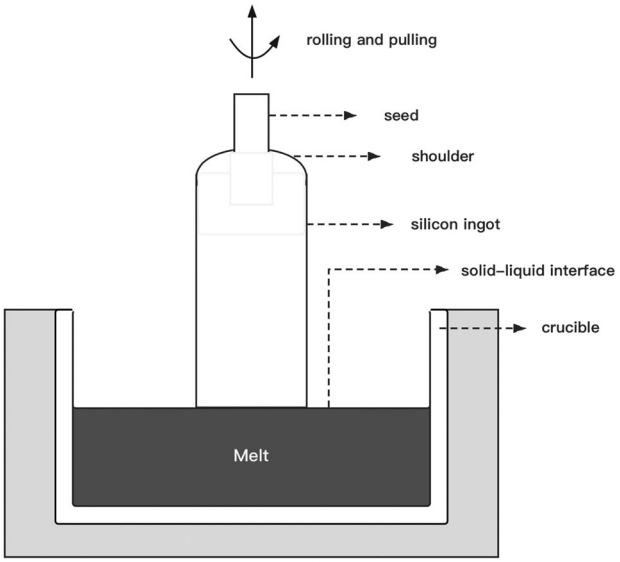


Figure 1. A simplified view of the CZ method as the monocrystalline silicon grows from the melt.

A motivating example of the monocrystalline silicon ingot growth process is considered. Figure 1 shows a schematic drawing of this silicon growth process in the Czochralski (CZ) technique, by which more than 90 percent of silicon wafers in markets are produced (Mohamed Ariff, Hashmi, and Brabazon 2018). After melting the polycrystalline silicon into the crucible and dipping a crystallographically oriented monocrystalline silicon seed into the melt, the seed crystal and the crucible both rotated at specific speed levels to first increase the ingot diameter and turn to isodiametric growth (Seigneur et al. 2016). Numerous sensors are placed into the crucible to collect real-time observations of variables, such as temperature, velocity, rotation, and pulling speeds of the ingot. These related variables can be divided into several subsystems, such as mechanics, thermal field, ingots, and gas supply, in which the variables can form a graph structure (Drouiche et al. 2015; Zhang et al. 2011). In addition, variables can also be classified by how they originated: (1) setting variables are those which can be changed automatically or artificially during the growth process; (2) measured variables are those which cannot be changed and contain most of the essential quality variables, such as ingot diameter measurement; (3) derivative variables are those whose information is contained in the former two groups of variables and can be directly calculated

by them similarly to taking the difference of two recent observations from a specific variable. It is essential to estimate and monitor the relationship between variables because even small shifts from the controlled state can affect the quality of the whole ingot at a great cost. However, by taking the traditional settings in DGMs that one node represents only one variable, the efficiency and accuracy of modeling and monitoring cannot be assured because of the great number of variables and neglection of prior expert knowledge. Therefore, a graph model that can simplify the variable structure while keeping the important relationship among variables is needed to address this problem by considering a subsystem as a node containing several variables.

In this article, to address the aforementioned problems, a novel directed graphical model is proposed, namely, the extended directed graphical model (EDGM). For real-world cases, such as the monocrystalline silicon growth process, with a large number of sensors and complex relationships between variables, the proposed EDGM methods have advantages over the existing methodologies in the following aspects: (1) they utilize directed graphical models to depict relationships between variables with flexibility structures of complex systems; (2) they separate variables into groups and consider one group as a node in a graph structure to decrease the complexity of the graph structure; (3) they use a hidden state layer to realize variable selection and model the functional relationship between the nodes; (4) they jointly model multiple essential quality outputs with a respective monitoring and diagnostic framework. In this work, an adjusted structure learning method is also proposed to extract the graph structure, and a monitoring and diagnosis framework is presented to capture system changes and help improve quality. This work is inspired by Yan et al.'s work, where they proposed a deep multitask learning method to predict quality outputs in an MMS (Yan et al. 2021). Similar to their work, hidden state vectors are introduced in the proposed work, and parameters are estimated by a stochastic proximal gradient descent method. The main difference is that in this work we have broadened the influential structure of variables from sequential to graphical, which suits complex manufacturing systems with more flexibility. We have also developed a structure learning method in case the graphical structure is unknown.

The remainder of this article is organized as follows. In Section 2, the proposed model's respective estimation, monitoring, and structure learning procedures are presented in detail. In Sections 3 and 4, examples, including numerical experiments and a real-world application to the monocrystalline silicon growth process, are provided to verify the efficiency of the proposed model. Finally, summaries and discussions about the model are presented in Section 5.

2. Model

In this section, problem settings are first formulated in Section 2.1, and then the proposed extended directed graphical model is introduced in Section 2.2. Parameter estimation steps are shown in Section 2.3, while the monitoring and diagnosis framework is introduced in Section 2.4. Finally, the learning steps of the graph structure are discussed in Section 2.5.

2.1. Problem formulation

A complex system that consists of a subsystem set or component set, $\mathcal{N} = \{1, 2, ..., K\}$, where K is the number of subsystems, that is, the number of nodes in the graph structure is considered. In this article, subsystems, components, and nodes in graphical models refer to the same meaning. Each subsystem contains multiple variables that can be separated into two groups: input variables and output variables. In real-world applications, the rules for separating the variables can be adjusted flexibly, for example, in-process sensing variables are considered as inputs and essential quality sensing variables as outputs. Taking the example of the monocrystalline silicon growth process, the most important quality variable is the silicon ingots' diameter and workers would check it during the whole process to assure the ingot has an ideal shape. Besides, there are many other variables that would influence the diameter, such as temperature in the melt, rolling and pulling speed of the seed. In this case, the ingot's diameter can be considered as the quality sensing variable, that is, output variable, and temperature, pulling, and rolling speed can be considered as in-processing sensing variables, that is, input variables.

We denote $\boldsymbol{x}_k = \{x_{k1}, x_{k2}, ..., x_{kn_{xk}}\}$ and $\boldsymbol{y}_k = \{y_{k1}, y_{k2}, ..., y_{kn_{yk}}\}$ as the input vector and output vector from the kth node, respectively, where n_{xk} and n_{yk} are the number of input variables and output variables, respectively. The subsystems interact with each other and form a graph structure $\mathcal{G}$, which contains a node set and an arc set, $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$, where $\mathcal{N} = \{1, 2, ..., K\}$ and the elements in $\mathcal{A}$ are node pairs. $(i, j) \in \mathcal{A}$ if an arc from node i to node j exists. $Pa(k)$ is denoted as the parent node set of node k, that is, $Pa(k) = \{j \in \mathcal{N} | (j, k) \in \mathcal{A}\}$. In this article, we further assume that there is no directed cycle in the graph,

which is a directed acyclic graph (DAG). Based on these problem settings, the goal of this article is to model the relationship between nodes that are attached to the arcs and predict the output variables $\{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_K\}$ based on the input variables $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_K\}$ and the functional relationships. DGM learning often includes two parts: estimating the relationship attached to the arcs between the nodes and learning the graph structure, that is, determining whether there is an arc between any pair of nodes. In Sections 2.2 and 2.3, we assume that the graph structure $\mathcal{A}$ is known and only estimate the relationship between node pairs in $\mathcal{A}$. In Section 2.5, this assumption is relaxed so that we need to decide the elements in $\mathcal{A}$ and estimate the respective relationship.

2.2. The extended directed graphical model

To realize the variable selection of input variables and the relationship between nodes, we introduce the hidden state vector $\boldsymbol{h}_k \in \mathbb{R}^{n_h}$ into the proposed EDGM, which represents the state of the subsystems. A simple instance of the EDGM is shown in Figure 2, which contains three nodes and two arcs, that is, $\mathcal{G}^{eg} = \{\mathcal{N}^{eg} = \{1,2,3\}, \mathcal{A}^{eg} = \{(1,2),(1,3)\}\}$.

As shown in the instance, we make the following assumptions regarding the proposed EDGM: (1) the graph consists of several nodes and $\mathcal{N}$ is the node set. It means that the whole system can be divided into several subsystems in practice, which is very common in a complex system. In the monocrystalline silicon growth process, the whole system can be divided into subsystems of heater, pressure field, and gas control, etc.; (2) for each node $k \in \mathcal{N}$, it consists of three parts: the input part $\boldsymbol{x}_k$, the hidden state part $\boldsymbol{h}_k$ and the output part $\boldsymbol{y}_k$. In practice, variables from a subsystem can be grouped as essential quality variables, that is, output variables, and process variables that might impact on the outputs, that is, input variables. The hidden state variables can be considered as the representation of the conditions of the subsystems, which cannot be directly observed; (3) the relationship within node k includes the functional relationship from $\boldsymbol{x}_k$ to $\boldsymbol{h}_k$ and from $\boldsymbol{h}_k$ to $\boldsymbol{y}_k$, which implies that the impact from inputs to outputs is realized by the hidden state variables. This assumption describes the real-world situation where the hidden condition of the subsystems is a reflection on the process variables and the hidden conditions decide whether the quality outputs are under control; (4) the relationship between nodes (if there is an arc between them) is realized between the hidden state variables of the two nodes. This assumption is based on the situation where not all input variables can impact outputs from other subsystems and only conditions or certain combinations of input variables can affect the other subsystems' conditions. According to the assumptions mentioned above, the relationship between input variables $\boldsymbol{x}_j$ and output variables $\boldsymbol{y}_k$ consists of three steps: firstly, $\boldsymbol{x}_j$ impacts on $\boldsymbol{h}_j$; secondly, $\boldsymbol{h}_j$ impacts on $\boldsymbol{h}_k$ if there exists a path from j to k according to the graphical structure; thirdly, $\boldsymbol{h}_k$ impacts on $\boldsymbol{y}_k$. If there is no path from j to k, then $\boldsymbol{x}_j$ has no impact on $\boldsymbol{y}_k$.

In this article, two types of parametric functions are assumed to model the relationship within each node and between nodes, respectively. The transmission functions describe how the hidden state vectors $\boldsymbol{h}_k$ are influenced, which is given as follows:

$$\boldsymbol{h}_k = f_k\left(\boldsymbol{x}_k, \{\boldsymbol{h}_j\}_{j \in Pa(k)}; \boldsymbol{\theta}_k^h\right) + \boldsymbol{z}_k, \quad k \in N \qquad [3]$$

where $\boldsymbol{\theta}_k^h$ is the model parameters from transmission functions and $\boldsymbol{z}_k$ is the transmission noise vector. On the other hand, the emission functions describe how the output variables $\boldsymbol{y}_k$ are influenced by the hidden state vector $\boldsymbol{h}_k$, which is given as follows:

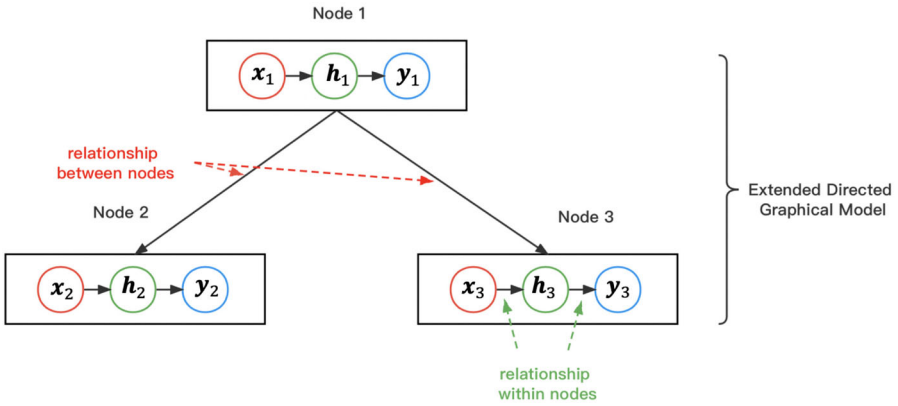


Figure 2. Graph structure and relationship between the nodes of the proposed model.

$$\boldsymbol{y}_k = g_k\left(\boldsymbol{h}_k; \boldsymbol{\theta}_k^g\right) + \boldsymbol{\epsilon}_k \quad [4]$$

where $\boldsymbol{\theta}_k^g$ is the model parameters from emission functions and $\boldsymbol{\epsilon}_k$ is the emission noise vector. According to the two types of functions, output variables are only directly affected by the hidden state vectors of the same node, while the hidden state vectors are affected by the inputs of the same node and hidden state vectors from the parent nodes. Usually, $n_{xk} \gg n_h$, and we assume all hidden state vectors $\boldsymbol{h}_k$s to have the same dimensions to realize variable selection and dimensional reduction of the input variables. For simplicity, we first assume linear forms of the two types of functions and normally distributed noises. Then, the model Eqs. [3] and [4] can be rewritten in linear forms as follows:

$$\begin{aligned}\boldsymbol{h}_k &= f_k\left(\boldsymbol{x}_k, \left\{\boldsymbol{h}_j\right\}_{j\in Pa(k)}; \boldsymbol{\theta}_k^h\right) + \boldsymbol{z}_k \\ &= \boldsymbol{W}_k\boldsymbol{x}_k + \sum_{j\in Pa(k)} \boldsymbol{U}_{jk}\boldsymbol{h}_j + \boldsymbol{z}_k\end{aligned} \quad [5]$$

$$\boldsymbol{y}_k = g_k\left(\boldsymbol{h}_k; \boldsymbol{\theta}_k^g\right) + \boldsymbol{\epsilon}_k = \boldsymbol{V}_k\boldsymbol{h}_k + \boldsymbol{\epsilon}_k \quad [6]$$

where $\boldsymbol{\theta}_k^h = \left\{\boldsymbol{W}_k, \{\boldsymbol{U}_{jk}\}_{j\in Pa(k)}\right\}$ and $\boldsymbol{\theta}_k^g = \{\boldsymbol{V}_k\}$; $\boldsymbol{z}_k \sim N(0, \sigma_z^2\boldsymbol{I}_z)$ and $\boldsymbol{\epsilon}_k \sim N\left(0, \sigma_\epsilon^2\boldsymbol{I}_\epsilon\right)$; $\boldsymbol{I}_z$ and $\boldsymbol{I}_\epsilon$ are identity matrices with respective dimensions. Based on the mathematical descriptions above, the goal of the model is: (1) estimating the model's parameters $\left\{\boldsymbol{\theta}_k^h, \boldsymbol{\theta}_k^g\right\}_{k\in\mathcal{N}}$ given the available dataset $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$, where $\mathcal{X} = \{\boldsymbol{X}_i\}_{i=1}^N$, $\mathcal{Y} = \{\boldsymbol{Y}_i\}_{i=1}^N$ and N is the number of observations. $\boldsymbol{X}_i = \left(\boldsymbol{x}_{i1}^T, \boldsymbol{x}_{i2}^T, \ldots, \boldsymbol{x}_{iK}^T\right)^T$ and $\boldsymbol{Y}_i = \left(\boldsymbol{y}_{i1}^T, \boldsymbol{y}_{i2}^T, \ldots, \boldsymbol{y}_{iK}^T\right)^T$ are the respective input vector and output vector from all nodes of the i th observation, $i = 1, \ldots, N$; (2) constructing a monitoring framework based on the model and providing out-of-control (OC) signals when a shift occurs; and (3) developing a structure learning and parameter estimation method when $Pa(j)$ is unknown. The realization of these goals will be explained in the following subsections.

2.3. Estimation of model parameters

The objective function for optimizing the relevant model parameters is inspired by the previous work and is given as follows (Yan et al. 2021):

$$\min_\Theta \mathcal{Q}(\Theta) = \mathcal{L}(\Theta) + \mathcal{R}(\Theta) \quad [7]$$

where $\Theta = \left\{\boldsymbol{\theta}_k^h, \boldsymbol{\theta}_k^g\right\}_{k\in\mathcal{N}}$ is the parameter set. Eq. [7] contains the prediction error loss term $\mathcal{L}(\Theta)$ and the regularization term for parameters, $\mathcal{R}(\Theta)$. For the loss function $\mathcal{L}(\Theta)$, the specific form is listed in Eq. [8] which is derived from the model assumptions.

$$L(\Theta) = L(\Theta; \mathcal{X}, \mathcal{Y}) = -\sum_{i=1}^N\sum_{k=1}^K \log P\left(\boldsymbol{y}_{ik}|\boldsymbol{h}_{ik}; \Theta\right) \quad [8]$$

where the loss function is in the form of the negative joint log-likelihood functions of output variables, given the model parameter set Θ and hidden state vectors $\boldsymbol{h}_{ik}$ s. Furthermore, we assume that the output variables in a node are conditionally independent of each other given the hidden state. Therefore, Eq. [6] can be rewritten as follows:

$$\begin{aligned}L(\Theta) = L(\Theta; \mathcal{X}, \mathcal{Y}) &= -\sum_{i=1}^N\sum_{k=1}^K \log P\left(\boldsymbol{y}_{ik}|\boldsymbol{h}_{ik}; \Theta\right) \\ &= -\sum_{i=1}^N\sum_{k=1}^K\sum_{j=1}^{n_{y,k}} \log P\left(y_{ikj}|\boldsymbol{h}_{ik}; \Theta\right)\end{aligned} \quad [9]$$

$\boldsymbol{e}_{ik} = \boldsymbol{y}_{ik} - g_k(\boldsymbol{h}_{ik}; \Theta)$ is denoted as the prediction error vector and obtains the following:

$$\begin{aligned}L(\Theta; \mathcal{X}, \mathcal{Y}) &= -\sum_{i=1}^N\sum_{k=1}^K \log P\left(\boldsymbol{y}_{ik}|\boldsymbol{h}_{ik}; \Theta\right) \\ &\propto \sum_{i=1}^N\sum_{k=1}^K L_k(\boldsymbol{e}_{ik}; \Theta)\end{aligned} \quad [10]$$

where $L_k(\boldsymbol{e}_{ik}; \Theta)$ is the negative log-likelihood function of the error vectors. There are multiple choices of the forms of $L_k(\boldsymbol{e}_{ik}; \Theta)$ in existing works, such as squared loss (Bukkapatnam et al. 2006) and Huber loss (Liu, Zhu, and Ma 2022). Usually, the choice is based on the assumed output distribution. In this article, the squared loss is selected for the Gaussian distributed noises as follows:

$$L_k(\boldsymbol{e}_{ik}; \Theta) = \|\boldsymbol{e}_{ik}\|^2 = \sum_{j=1}^{n_{y,k}} e_{ikj}^2 \quad [11]$$

When the graph structure $\mathcal{A}$ is known, the regularization function is listed in Eq. [10] as follows:

$$\mathcal{R}(\Theta) = \frac{\lambda_R}{2}\|\Theta\|^2 \quad [12]$$

where $\|\cdot\|^2$ is the L_2 penalty for matrices, that is, sum of squares of all elements in the parameter set. When $\mathcal{A}$ is unknown, some adjustments are added to the regularization function, which is discussed in Section 2.5.

The stochastic proximal gradient algorithm is selected as the optimization step for Eq. [5]. Without loss of generality, we assume the size of a minibatch is 1 in the following updating equations, that is, $\mathcal{D}^m =$

$\{\boldsymbol{x}, \boldsymbol{y}\}$. For convenience, $\mathcal{C}(k_a, k_b)$ is denoted as the set of all possible chains from node k_a to node k_b. A chain $c = (k_a, k_l \cdots, k_m, k_b) \in \mathcal{C}(k_a, k_b)$ if all arcs in sequentially paired nodes of c exist in $\mathcal{A}$, that is, $\{(k_a, k_l), ..., (k_m, k_b)\} \subseteq \mathcal{A}$. Given $c \in \mathcal{C}(k_a, k_b)$, $d^c(k_a, k_b)$ is denoted as the gradient chain of hidden state vectors from node k_a to node k_b according to c as follows:

$$d^c(k_a, k_b) = \frac{\partial \boldsymbol{h}_{k_b}}{\partial \boldsymbol{h}_{k_m}} \cdot \cdots \cdot \frac{\partial \boldsymbol{h}_{k_l}}{\partial \boldsymbol{h}_{k_a}} \quad [13]$$

$Ac(k)$ is denoted as the set of nodes that are influenced by the parameters of node k, that is, $Ac(k) = \{l \in \mathcal{N}, l \neq k | \mathcal{C}(k,l) \neq \emptyset\} \cup \{k\}$. Note that we set $k \in Ac(k)$, $\forall k \in \mathcal{N}$ and definite $d^c(k,k) = \boldsymbol{I}_{d_h}$, $|\mathcal{C}(k,k)| = 1$ as a special case for convenience. Based on these notations, the gradient from $\mathcal{L}(\Theta)$ to parameters Θ can be given in Eqs. [14] and [15].

For transmission function parameters:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{\theta}_k^h} \mathcal{L}(\boldsymbol{x}, ; \Theta) &= -\sum_{l \in \mathcal{N}} \frac{\partial}{\partial \boldsymbol{\theta}_k^h} \log P(\boldsymbol{y}_l | \boldsymbol{h}_l, \boldsymbol{\theta}_l^g) \\ &= -\frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{\theta}_k^h} \sum_{l \in Ac(k)} \left\{ \frac{\partial}{\partial \boldsymbol{h}_l} \log P(\boldsymbol{y}_l | \boldsymbol{h}_l, \boldsymbol{\theta}_l^g) \right\} \\ = \frac{\partial \boldsymbol{h}_k}{\partial \boldsymbol{\theta}_k^h} \sum_{l \in Ac(k)} \left\{ \left[\sum_{c \in \mathcal{C}(k,l)} d_h^c(k,l) \right] \cdot 2\left(g_k(\boldsymbol{h}_k, \boldsymbol{\theta}_k^g) - \boldsymbol{y}_k\right) \cdot \frac{\partial g_k(\boldsymbol{h}_k, \boldsymbol{\theta}_k^g)}{\partial \boldsymbol{h}_k} \right\} \end{aligned} \quad [14]$$

For emission function parameters:

$$\frac{\partial}{\partial \boldsymbol{\theta}_k^g} L(\boldsymbol{x}, \boldsymbol{y}; \Theta) = -\frac{\partial}{\partial \boldsymbol{\theta}_k^g} \log P(\boldsymbol{y}_k | \boldsymbol{h}_k, \boldsymbol{\theta}_k^g) = 2\left(g_k(\boldsymbol{h}_k, \boldsymbol{\theta}_k^g) - \boldsymbol{y}_k\right) \cdot \frac{\partial g_k(\boldsymbol{h}_k, \boldsymbol{\theta}_k^g)}{\partial \boldsymbol{\theta}_k^g} \quad [15]$$

On the other hand, the gradient from $\mathcal{R}(\Theta)$ to parameters Θ is given as follows:

$$\frac{\partial}{\partial \Theta} R(\Theta) = \lambda_R \Theta \quad [16]$$

In this work, the stochastic proximal gradient algorithm is used to estimate the parameter set Θ, where the updating equation for parameters from iteration $(\tau - 1)$ to τ is shown in Eq. [17] as follows:

$$\Theta^{(\tau)} = \Theta^{(\tau-1)} - \eta \left\{ \left[\frac{\partial}{\partial \Theta} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y}; \Theta) + \frac{\partial}{\partial \Theta} \mathcal{R}(\Theta) \right] | \Theta = \Theta^{(\tau-1)} \right\} \quad [17]$$

where η is the learning rate. With an initial value of parameters, $\Theta^{(0)}$ and specific forms of functions, the parameters can be optimized by recursively calculating Eq. [17] based on Eqs. [14] to [16] until convergence. In each iteration, we update all the parameters in Θ by the updating equation, if $\theta \in \Theta$ is from the transmission parameter set, that is, $\theta \in \boldsymbol{\theta}^h$, we use the updating equation replacing $\frac{\partial}{\partial \Theta} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y}; \Theta)$ and $\frac{\partial}{\partial \Theta} \mathcal{R}(\Theta)$ by Eqs. [14] and [16] in the manuscript, respectively. On the other hand, if $\theta \in \Theta$ is from the emission parameter set, that is, $\theta \in \boldsymbol{\theta}^g$, we use the updating equation replacing $\frac{\partial}{\partial \Theta} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y}; \Theta)$ and $\frac{\partial}{\partial \Theta} \mathcal{R}(\Theta)$ by Eqs. [15] and [16] in the manuscript, respectively. Detailed steps of the algorithm is summarized in Algorithm 1.

Algorithm 1: Stochastic proximal gradient descent algorithm for parameter estimation

Input: The training set $\left\{ \{\boldsymbol{x}_{ik}, \boldsymbol{y}_{ik}\}_{k=1}^K \right\}_{i=1}^{N_{train}}$; the validation set $\left\{ \{\boldsymbol{x}_{ik}, \boldsymbol{y}_{ik}\}_{k=1}^K \right\}_{i=1}^{N_{valid}}$; the tuning set for λ_R, Λ_R; the learning rate η; the recursion limit parameter ξ

Output: Estimated parameter set $\hat{\Theta} = \left\{ \hat{\boldsymbol{\theta}}_k^h, \hat{\boldsymbol{\theta}}_k^g \right\}_{k=1}^K$

(1) Initialize $\Theta^{(0)} = 0$
(2) **for** $\lambda_R \in \Lambda_R$
(3) Set $\tau = 0, \Theta^{(0)}(\lambda_R) = \Theta^{(0)}$
(4) **do**
(5) **for** $k = \{1, ..., K\}$
(6) Randomly select $i \in \{1, ..., N_{train}\}$ and get $\{\boldsymbol{x}_{ik}, \boldsymbol{y}_{ik}\}$
(7) **for** $\theta \in \Theta^{(\tau)}(\lambda_R)$
(8) **if** $\theta \in \boldsymbol{\theta}^{h(\tau)}(\lambda)$:
(9) Update $\theta^{(\tau+1)}(\lambda_R)$ by Eq. [17] replacing $\frac{\partial}{\partial \Theta} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y}; \Theta)$ and $\frac{\partial}{\partial \Theta} \mathcal{R}(\Theta)$ by Eqs. [14] and [16]
(10) **else**
(11) Update $\theta^{(\tau+1)}(\lambda_R)$ by Eq. [17] replacing $\frac{\partial}{\partial \Theta} \mathcal{L}(\boldsymbol{x}, \boldsymbol{y}; \Theta)$ and $\frac{\partial}{\partial \Theta} \mathcal{R}(\Theta)$ by Eqs. [15] and [16]
(12) **end if**
(13) **end for**
(14) **end for**
(15) Set $\tau \leftarrow \tau + 1$
(16) **until** $\|\Theta^{(\tau)}(\lambda_R) - \Theta^{(\tau-1)}(\lambda_R)\|^2 < \xi$
(17) **end do**
(18) Set $\hat{\Theta}(\lambda_R) = \Theta^{(\tau)}(\lambda_R)$
(19) Calculate RMSE using parameter $\hat{\Theta}(\lambda_R)$ in validation set getting $RMSE(\lambda_R)$
(20) **end for**
(21) The final tuning parameter is set to be $\hat{\lambda}_R = \arg\min_{\lambda_R \in \Lambda_R} RMSE(\lambda_R)$
(22) The final estimated parameter set is set to be $\hat{\Theta} = \hat{\Theta}(\hat{\lambda}_R)$

Although the global optimum property is not assured in this algorithm, restarting with another randomized initial value will considerably increase its effectiveness in real cases when validation accuracy is unsatisfactory (Yan et al. 2021). Note that the estimation steps [7] to [17] have no limitations on the function forms $\boldsymbol{h}_k =$

$f_k\left(\boldsymbol{x}_k, \{\boldsymbol{h}_j\}_{j\in Pa(k)}; \boldsymbol{\theta}_k^h\right)$ and $\boldsymbol{y}_k = g_k\left(\boldsymbol{h}_k; \boldsymbol{\theta}_k^g\right)$. Although a linear assumption is given in this article for EDGM in Eqs. [5] and [6], nonlinear extensions can be easily made by changing the specific forms of $f_k(\cdot)$ and $g_k(\cdot)$.

2.4. Monitoring and diagnosis framework of the EDGM

In this subsection, a phase II monitoring method and a diagnosis framework are introduced based on the estimated model parameters and the known graph $\mathcal{G} = \{\mathcal{N}, \mathcal{A}\}$. In the EDGM, it is assumed that the inputs and outputs $\boldsymbol{x}_k$s and $\boldsymbol{y}_k$s can be observed directly, and many existing control charts have proven to be effective when a mean shift occurs in some of the input and output variables, such as CUSUM and multivariate exponentially weighted moving average (MEWMA) methods. In this article, we mainly focus on how to effectively monitor the relationship change in the EDGM, where the input variables might distribute the same when the system is out of control (OC) with no significant shifts in output variables. The OC model of the EDGM only differs from the in-control (IC) model in hidden state vectors by assuming that a mean shift occurs in transmission functions, which is given as follows if a shift occurs at node k^{OC}:

$$\boldsymbol{h}_k^{OC} = f_k^{OC}\left(\boldsymbol{x}_k, \{\boldsymbol{h}_j\}_{j\in Pa(k)}; \boldsymbol{\theta}_k\right) = \boldsymbol{W}_k\boldsymbol{x}_k + \sum_{j\in Pa(k)} \boldsymbol{U}_{jk}\boldsymbol{h}_j + \boldsymbol{z}_k + \boldsymbol{s}_k \quad [18]$$

where $\boldsymbol{s}_k = \delta\boldsymbol{1}_k$ is the shift vector with shift scale δ. The monitoring method in this article is inspired by the directional multivariate exponentially weighted moving average (DMEWMA) method, which aggregates the shift scale by taking advantage of the sequentially affected structure of multistage manufacturing systems (Zou and Tsung 2008). $Apa(k)$ is denoted as the set of all nodes that have influence on node k, directly or intermediately, that is, $Apa(k) = \{j \in \mathcal{N}, j \neq k | \mathcal{C}(j,k) \neq \emptyset\} \cup \{k\}$; $\Phi^c(k_a, k_b)$ is denoted as the hidden state transition coefficient matrix from node k_a to node k_b according to the chain $c = (k_a, k_l \cdots, k_m, k_b) \in \mathcal{C}(k_a, k_b)$ as:

$$\Phi^c(k_a, k_b) = \boldsymbol{U}_{k_a k_l} \cdot \cdots \cdot \boldsymbol{U}_{k_m k_b} \quad [19]$$

Given the known graph structure, the EDGM [3] to [4] can be rewritten without the hidden state vectors as follows:

$$\boldsymbol{y}_k = \sum_{j\in Apa(k)} \sum_{c\in\mathcal{C}(j,k)} \boldsymbol{V}_k \Phi^c(j,k) \boldsymbol{W}_j \boldsymbol{x}_j + \boldsymbol{\varepsilon}_k \quad [20]$$

$$\boldsymbol{\varepsilon}_k = \sum_{j\in Apa(k)} \sum_{c\in\mathcal{C}(j,k)} \boldsymbol{V}_k \Phi^c(j,k) \boldsymbol{z}_j + \boldsymbol{\epsilon}_k \quad [21]$$

According to Eqs. [20] and [21], a vector regression containing variables from all nodes can be shown as:

$$y = \begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_K \end{bmatrix} = \boldsymbol{\Psi} \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_K \end{bmatrix} + \boldsymbol{\Gamma} \begin{bmatrix} \boldsymbol{z}_1 \\ \vdots \\ \boldsymbol{z}_K \end{bmatrix} + \begin{bmatrix} \boldsymbol{\epsilon}_1 \\ \vdots \\ \boldsymbol{\epsilon}_K \end{bmatrix} \quad [22]$$

where $\boldsymbol{\Psi}$ and $\boldsymbol{\Gamma}$ are $K \times K$ block matrices, $\boldsymbol{\Psi} = \{\Psi_{j,k}\}$ and $\boldsymbol{\Gamma} = \{\Gamma_{j,k}\}$; $\Psi_{j,k}$ and $\Gamma_{j,k}$ are submatrices in the jth row and kth column of $\boldsymbol{\Psi}$ and $\boldsymbol{\Gamma}$, respectively.

$$\Psi_{j,k} = \begin{cases} \sum_{c\in\mathcal{C}(j,k)} \boldsymbol{V}_k \Phi^c(j,k) \boldsymbol{W}_j, & j \in Apa(k) \\ \boldsymbol{0}, & otherwise \end{cases}, \quad \Gamma_{j,k} = \begin{cases} \sum_{c\in\mathcal{C}(j,k)} \boldsymbol{V}_k \Phi^c(j,k), & j \in Apa(k) \\ 0, & otherwise \end{cases} \quad [23]$$

Without loss of generality, we assume $\mathbb{E}\boldsymbol{y} = \boldsymbol{\mu}_y = 0$, and there is only one OC node at the same time. Since the possibilities of the OC node are limited, the hypothesis test under the OC model assumptions can be concluded as follows:

$$H_0: \boldsymbol{\mu}_y = 0, \ H_1: \boldsymbol{\mu}_y = \delta\boldsymbol{d}_1 \ or \ \boldsymbol{\mu}_y = \delta\boldsymbol{d}_2 \ or \cdots or \ \boldsymbol{\mu}_y = \delta\boldsymbol{d}_K$$

$$\boldsymbol{d}_k \propto \Gamma \begin{pmatrix} 0 \\ \vdots \\ 1_k \\ \vdots \\ 0 \end{pmatrix} \quad [24]$$

where δ is the expected shift scale. Eq. [24] uses information from the learned graph, which is contained in matrix $\boldsymbol{\Gamma}$, to identify how the shifts from hidden state vectors will affect output variables. Based on the Gaussian noise assumption, the general likelihood ratio test will lead to the determination of whether the following test statistic is larger than a specific threshold (Zou and Tsung 2008):

$$\max_{\boldsymbol{k}\in\mathcal{N}} \left|\boldsymbol{d}_k^T \boldsymbol{\Sigma}_y \boldsymbol{y}\right| \quad [25]$$

where $\boldsymbol{\Sigma}_y$ is the covariance matrix of the output vector of all nodes, $\boldsymbol{y}$. It is expected to outperform the basic T^2 test of $|\boldsymbol{y}^T\boldsymbol{\Sigma}_y\boldsymbol{y}|$ because Eq. [25] has more specific shift information. $\boldsymbol{z}_t^{\boldsymbol{y}} = (1-\lambda)\boldsymbol{z}_{t-1}^{\boldsymbol{y}} + \lambda\boldsymbol{y}_t$ is denoted as the statistic containing historical information, where $\boldsymbol{y}_t$ is the t th observation of the whole output vector $\boldsymbol{y}$

and $\lambda \in (0,1]$ is the smoothing parameter. Similar to DMEWMA, a graph-based multivariate exponentially weighted moving average (GB-MEWMA) is proposed to integrate the structural information of the graph, which is expressed as follows with a starting vector $\boldsymbol{z}_0^{\boldsymbol{y}}$:

$$MZ_t^{\boldsymbol{y}} = \max_{k\in\mathcal{N}}\left(\sqrt{\frac{\lambda}{2-\lambda}}\left|\boldsymbol{d}_k'\boldsymbol{\Sigma}_z^{-1}\boldsymbol{z}_t^{\boldsymbol{y}}\right|\right),\ \boldsymbol{\Sigma}_z = \frac{\lambda}{(2-\lambda)}\boldsymbol{\Sigma}_y \tag{26}$$

When $\boldsymbol{z}_0^{\boldsymbol{y}}$ is selected to be zero, $\mathbb{E}\boldsymbol{d}_k'\boldsymbol{\Sigma}_z^{-1}\boldsymbol{z}_t^{\boldsymbol{y}} = 0$ and $\mathrm{Var}\left(\boldsymbol{d}_k'\boldsymbol{\Sigma}_z^{-1}\boldsymbol{z}_t^{\boldsymbol{y}}\right) \to (2-\lambda)/\lambda$ can be derived under the IC model (Xiang and Tsung 2008). This GB-MEWMA raises an OC alarm when the charting statistic is larger than a threshold value, that is, $MZ_t^{\boldsymbol{y}} > L_G$, where L_G is determined by a prespecified IC average run length (ARL).

For the diagnosis framework of the GB-MEWMA, Eqs. [27] and [28] introduce a direct method for detecting the expected OC node $\hat{\kappa}_{OC}$ together with the starting OC observation index $\hat{\tau}_{OC}$ (Zou and Tsung 2008) as follows:

$$\hat{\kappa}_{OC} = \arg_{k\in\mathcal{N}}\max\left|\sqrt{\frac{\lambda}{2-\lambda}}\left|\boldsymbol{d}_k'\boldsymbol{\Sigma}_z^{-1}\boldsymbol{z}_m^{\boldsymbol{y}}\right|\right| \tag{27}$$

$$\hat{\tau}_{OC} = \arg_{0\le t<m}\max\left\{\frac{1}{m-t}\left(\sum_{i=t+1}^{m}\boldsymbol{y}_t{}'\right)\boldsymbol{\Sigma}_y^{-1}\left(\sum_{i=t+1}^{m}\boldsymbol{y}_t\right)\right\} \tag{28}$$

where $t = 0,1,\ldots,m$ are indices of all monitored observations. The change point OC model is assumed in this subsection, where the system follows the IC model at observations $t = 0,1,\ldots,\tau$, $(\tau < m)$, and follows the OC model at $t = \tau+1,\ldots,m$.

2.5. Structure learning of the EDGM

In the above subsections, it is assumed that a known graph structure, $\mathcal{G} = \{\mathcal{N},\mathcal{A}\}$, is given. However, there are real-world cases when we have to determine whether there is a relationship between specific pairs of nodes. In this subsection, when is unknown, a candidate parent node set $Cpa(k)$, $k \in \mathcal{N}$ is given to replace the actual parent node set $Pa(k)$ and we assume that $Pa(k) \subseteq Cpa(k)$. It is a very weak assumption since we can simply decide $Cpa(k)$ to be all other nodes that do not violate the DAG assumption.

The objective function for the EDGM's structure learning is given as follows:

$$\min_{\Theta}\mathcal{S}(\Theta) = \mathcal{L}_{\mathcal{S}}(\Theta) + \mathcal{R}_{\mathcal{S}}(\Theta) \tag{29}$$

Similar to the objective function in Eq. [7] for estimating parameters, it contains the loss function part and regularization part. The loss function $\mathcal{L}_{\mathcal{S}}(\Theta)$ is still the negative log-likelihood function of the prediction error, except that prediction of outputs is determined by the structure based on the candidate parent node set, as shown in Eqs. [30] and [31]:

$$\boldsymbol{h}_k = f_k\left(\boldsymbol{x}_k,\{\boldsymbol{h}_j\}_{j\in Cpa(k)};\boldsymbol{\theta}_k^h\right) + \boldsymbol{z}_k = \boldsymbol{W}_k\boldsymbol{x}_k + \sum_{j\in Cpa(k)}\boldsymbol{U}_{jk}\boldsymbol{h}_j + \boldsymbol{z}_k \tag{30}$$

$$\boldsymbol{y}_k = g_k\left(\boldsymbol{h}_k;\boldsymbol{\theta}_k^g\right) + \boldsymbol{\epsilon}_k = \boldsymbol{V}_k\boldsymbol{h}_k + \boldsymbol{\epsilon}_k \tag{31}$$

The error vector $\boldsymbol{e}_{ik} = \boldsymbol{y}_{ik} - g_k(\boldsymbol{h}_i;\Theta)$ is calculated by the above equations. The loss function $\mathcal{L}_{\mathcal{S}}(\Theta)$ has similar gradients as [14] and [15] with $Pa(k)$ replaced by $Cpa(k)$ and $\mathcal{C}(j,k)$ is adjusted based on $Cpa(k)$. For regularization function $\mathcal{R}_{\mathcal{S}}(\Theta)$, we add a group LASSO penalty term to penalize the coefficient matrix of an insignificant parent node to zero in all elements, which is given as follows:

$$\mathcal{R}_{\mathcal{S}}(\Theta) = \frac{\lambda_1}{2}\|\Theta\|^2 + \lambda_2\sum_{k\in\mathcal{N}}\sum_{j\in Cpa(k)}\sqrt{q_{jk}}\|\boldsymbol{U}_{jk}\|_F \tag{32}$$

where $\|\cdot\|_F$ is the Frobius norm of a matrix, that is, the square root of the sum of squares of all elements in a matrix and q_{jk} is the size of the matrix $\boldsymbol{U}_{jk}$, and $\{\lambda_1,\lambda_2\}$ are the respective penalty parameters. The penalty function [32] consists of two parts: original L_2 norm for parameter set Θ and group LASSO penalty, which have different objectives. The group sparse LASSO penalty tends to make all elements in several $\boldsymbol{U}_{jk}$ s, $j \in Cpa(k)$, equal to 0, which can realize the objective of selecting parent nodes for node k. On the other hand, for the selected parent nodes whose respective $\boldsymbol{U}_{jk} \neq 0$, we still want a sparse $\boldsymbol{U}_{jk}$ for the objective of identifying the important relationship between hidden state vectors. Note that the second term in Eq. [32] is not a convex function, and therefore, cannot be directly used in the stochastic proximal gradient algorithm framework. To transfer a nonconvex objective function into a convex objective function, many techniques have proven to be effective with satisfactory properties, such as local linear approximation (LLA) (Zou and Li 2008), smoothly clipped absolute deviation (SCAD) (Fan and Li 2001) and minimax concave penalty (MCP) (Zhang 2010). In this article, an adjusted form of the MCP is used by approximating $\mathcal{R}_{\mathcal{S}}(\Theta)$ by $\tilde{\mathcal{R}}_{\mathcal{S}}(\Theta)$ as follows:

$$\tilde{\mathcal{R}}_{\mathcal{S}}(\Theta) = \frac{\lambda_1}{2}\|\Theta\|^2 + \lambda_2 \sum_{k\in\mathcal{N}} \sum_{j\in Cpa(k)} p_{\lambda_2,\gamma}(\|\boldsymbol{U}_{jk}\|_F)$$

$$\frac{\partial p_{\lambda_2,\gamma}(\|\boldsymbol{A}\|_F)}{\partial \boldsymbol{A}} = \begin{cases} \lambda_2 1_{\boldsymbol{A}} - \dfrac{\boldsymbol{A}}{\gamma}, \|\boldsymbol{A}\|_F \le \lambda_2\gamma \\ 0, \ \|\boldsymbol{A}\|_F > \lambda_2\gamma \end{cases} \quad [33]$$

where $\gamma > 1$ is a tuning parameter for penalizing the coefficient matrix. As $\gamma \to \infty$, elements from all $\boldsymbol{U}_{jk}$s will turn to zero, resulting in a sparser graph structure. The details of the MCP and discussion on choosing hyperparameters can be found in previous works (Breheny and Huang 2011). Replacing $\mathcal{R}_{\mathcal{S}}(\Theta)$ with $\tilde{\mathcal{R}}_{\mathcal{S}}(\Theta)$, a similar stochastic proximal gradient algorithm in Section 2.3, can be utilized according to Eqs. [14], [15], [17], and [33]. Although there are many other effective methods of graph structure learning, Eq. [33] is used in this work because it suits the special structure of EDGM and can simultaneously address the problems of the structure learning step together with parameter estimation. Algorithm 2 provides detailed steps for the structure learning method.

Algorithm 2: Stochastic proximal gradient descent algorithm for structure learning

Input: The training set $\left\{\{\boldsymbol{x}_{ik}, \boldsymbol{y}_{ik}\}_{k=1}^{K}\right\}_{i=1}^{N_{train}}$; the validation set $\left\{\{\boldsymbol{x}_{ik}, \boldsymbol{y}_{ik}\}_{=1}^{K}\right\}_{i=1}^{N_{valid}}$; the tuning sets for λ_1, λ_2 and γ, are $\Lambda_1, \Lambda_2, \Lambda_\gamma$,; the learning rate η; the recursion limit parameter ξ; the candidate parent set $\{Cpa(k)\}_{k=1}^{K}$

Output: Estimated parameter set $\hat{\Theta} = \left\{\hat{\boldsymbol{\theta}}_k^h, \hat{\boldsymbol{\theta}}_k^g\right\}_{k=1}^{K}$; Estimated structure $\left\{\widehat{Pa(k)}\right\}_{k=1}^{K}$

(1) Initialize $\Theta^{(0)} = 0$, $Pa(k)^{(0)} = Cpa(k)$, $k = 1, ..., K$
(2) **for** $(\lambda_1, \lambda_2, \gamma) \in \Lambda_1 \times \Lambda_2 \times \Lambda_\gamma$
(3) Set $\tau = 0, \Theta^{(0)}(\lambda_1, \lambda_2, \gamma) = \Theta^{(0)}$ and $Pa(k)^{(0)}(\lambda_1, \lambda_2, \gamma) = Pa(k)^{(0)}$
(4) **do**
(5) **for** $k = \{1, ..., K\}$
(6) Randomly select $i \in \{1, ..., N_{train}\}$ and get $\{\boldsymbol{x}_{ik}, \boldsymbol{y}_{ik}\}$
(7) **for** $\theta \in \Theta^{(\tau)}(\lambda_1, \lambda_2, \gamma)$
(8) **if** $\theta \in \boldsymbol{\theta}^{h(\tau)}(\lambda_1, \lambda_2, \gamma)$:
(9) Update $\theta^{(\tau+1)}(\lambda_1, \lambda_2, \gamma)$ by Eq. [17] replacing $\frac{\partial}{\partial\Theta}\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}; \Theta)$ and $\frac{\partial}{\partial\Theta}\mathcal{R}(\Theta)$ by Eqs. [14] and [33], based on $Pa(k)^{(\tau)}(\lambda_1, \lambda_2, \gamma)$
(10) **else**
(11) Update $\theta^{(\tau+1)}(\lambda_1, \lambda_2, \gamma)$ by Eq. [17] replacing $\frac{\partial}{\partial\Theta}\mathcal{L}(\boldsymbol{x}, \boldsymbol{y}; \Theta)$ and $\frac{\partial}{\partial\Theta}\mathcal{R}(\Theta)$ by Eqs. [15] and [33], based on $Pa(k)^{(\tau)}(\lambda_1, \lambda_2, \gamma)$
(12) **end if**
(13) **end for**
(14) **end for**
(15) Set $\tau \leftarrow \tau + 1$
(16) **until** $\|\Theta^{(\tau)}(\lambda_1, \lambda_2, \gamma) - \Theta^{(\tau-1)}(\lambda_1, \lambda_2, \gamma)\|^2 < \xi$
(17) **end do**
(18) Set $\hat{\Theta}(\lambda_1, \lambda_2, \gamma) = \Theta^{(\tau)}(\lambda_1, \lambda_2, \gamma)$
(19) **for** $k = 1, ..., K$
(20) **for** $j \in Pa(k)^{(\tau)}(\lambda_1, \lambda_2, \gamma)$
(21) **if** $\boldsymbol{U}_{jk}^{(\tau)}(\lambda_1, \lambda_2, \gamma) = 0$:
(22) Remove j from $Pa(k)^{(\tau)}(\lambda_1, \lambda_2, \gamma)$
(23) Set $\widehat{Pa(k)}(\lambda_1, \lambda_2, \gamma) = Pa(k)^{(\tau)}(\lambda_1, \lambda_2, \gamma)$
(24) Calculate RMSE using parameter $\hat{\Theta}(\lambda_1, \lambda_2, \gamma)$ and $\widehat{Pa(k)}(\lambda_1, \lambda_2, \gamma)$ in validation set getting $RMSE(\lambda_1, \lambda_2, \gamma)$
(25) **end for**
(26) The final combination is set to be $(\hat{\lambda}_1, \hat{\lambda}_2, \hat{\gamma}) = \arg\min_{(\lambda_1,\lambda_2,\gamma)\in\Lambda_1\times\Lambda_2\times\Lambda_\gamma} RMSE(\lambda_1, \lambda_2, \gamma)$
(27) The final estimated parameter set is set to be $\hat{\Theta} = \hat{\Theta}(\hat{\lambda}_R)$
(28) The final estimated structure is set to be $\widehat{Pa(k)} = \widehat{Pa(k)}(\hat{\lambda}_1, \hat{\lambda}_2, \hat{\gamma}),\ k = 1, ..., K$

There are many hyperparameters to be selected in former subsections: (1) we set the dimensions of the hidden state vector, $\boldsymbol{h}_k$ to be the same, that is, n_h, to control the expected information that can influence output variables from the inputs, which can be determined by real-world cases or dimension reduction techniques; (2) penalty parameters $\lambda_R, \lambda_1, \lambda_2$ are tuned from a given candidate parameter set by the root mean square error (RMSE) of validation observations.

3. Numerical experiments

In this section, numerical simulation studies are conducted to verify the effectiveness of the proposed EDGM along with the following techniques discussed in Section 2: parameter estimation, monitoring framework, and graph structure learning.

Similar to previous works (Yan et al. 2021), the simulation data in this section are generated according to the following equations:

$$\boldsymbol{h}_{ik} = \boldsymbol{W}_k \boldsymbol{x}_{ik} + \sum_{j\in Pa(k)} \boldsymbol{U}_{jk}\boldsymbol{h}_{ij} + \boldsymbol{z}_{ik} \quad [34]$$

$$\boldsymbol{y}_{ik} = \boldsymbol{V}_k \boldsymbol{h}_{ik} + \boldsymbol{\epsilon}_{ik} \quad [35]$$

where $i = 1, ..., N$ are the index of observations; the transition error $\boldsymbol{z}_{ik}$ and emission error $\boldsymbol{\epsilon}_{ik}$ follow a Gaussian distribution, $\boldsymbol{z}_k \sim N(0, \sigma_z^2 \boldsymbol{I}_z)$ and $\boldsymbol{\epsilon}_k \sim N(0, \sigma_\epsilon^2 \boldsymbol{I}_\epsilon)$ where $\boldsymbol{I}_z$ and $\boldsymbol{I}_\epsilon$ are identity matrices with respective dimensions; each element in $\boldsymbol{x}_{ik}$ is

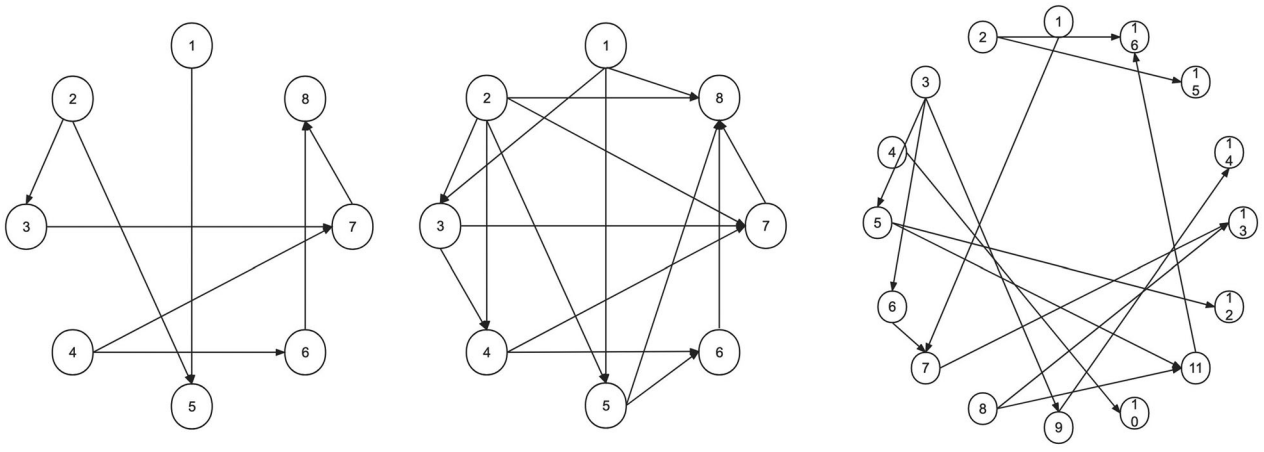


Figure 3. The three designed graphs: (a) G1 with 8 nodes and 8 arcs; (b) G2 with 8 nodes and 16 arcs; (c) G3 with 16 nodes and 16 arcs.

Table 1. Factor settings for simulation studies.

Factors	Levels	Meanings
Nodes	Level 1: 8 Level 2: 16	Number of nodes in the graph
Density	Level 1: 1 Level 2: 2	Proportion of # of arcs to # of nodes in the graph
Sample size	Level 1: 10 Level 2: 100	Training sample size
Emission Noise	Level 1: 0.1 Level 2: 1	Variance of noise in outputs
Transition Noise	Level 1: 0.1 Level 2: 1	Variance of noise in hidden state

generated from $N(0,1)$; each element in $\boldsymbol{W}_k$ is generated from $N\left(0, \frac{1}{\sqrt{n_{xk}}}\right)$; and each element in $\boldsymbol{U}_{jk}$, $\boldsymbol{V}_k$ is generated from $N\left(0, \frac{1}{\sqrt{n_h}}\right)$ where n_{xk} and n_h are dimensions of input variables of node k and hidden state vectors, respectively. In this article, we set $n_{xk} = n_x = 20$, $n_h = 3$ and $n_{yk} = n_y = 5$ in the later simulation studies. There are three graph structures that are used in this section, which are denoted as G1, G2, and G3, and represent three types of scenarios: a small number of nodes with sparse arcs; a small number of nodes with dense arcs; and a large number of nodes with sparse arcs. The specific structures of the three designed graphs are shown in Figure 3.

In the first part, we will verify the effectiveness of the EDGM by evaluating the prediction error of outputs in test data. Three other models are compared as benchmarks: (1) do not separate the inputs and outputs and denote $\boldsymbol{z}_k^T = [\boldsymbol{x}_k^T, \boldsymbol{y}_k^T]$. The respective graphical model is given as follows:

$$\boldsymbol{z}_{ik} = \sum_{j \in Pa(k)} \boldsymbol{C}_{jk}\boldsymbol{z}_{ij} + \boldsymbol{\varepsilon}_{ik}, \quad k \in N \qquad [36]$$

which is named as Benchmark 1 where $\boldsymbol{\varepsilon}_{ik}$ are white noises; (2) there is a separation of inputs and outputs but do not introduce hidden state vectors. The respective graphical model is given as follows:

$$\boldsymbol{y}_{ik} = \boldsymbol{D}_{kk}\boldsymbol{x}_{ik} + \sum_{j \in Pa(k)} \boldsymbol{D}_{jk}\boldsymbol{x}_{ij} + \boldsymbol{\varepsilon}_{ij}, \quad k \in N \qquad [37]$$

which is named as Benchmark 2; (3) for a given node $k \in \mathcal{N}$, select all chains that end with k, that is, $\mathcal{C}^A(k) = \{\mathcal{C}(j,k) | j \in \mathcal{N}, j \neq k\}$. Based on each chain $c_k \in \mathcal{C}^A(k)$, a traditional state space model can be constructed with the sequential substructure and obtain a prediction of outputs $\hat{\boldsymbol{y}}_{ik}^{(c_k)}$. The final prediction is calculated as the mean of predictions from all chains as follows:

$$\hat{\boldsymbol{y}}_{ik}^{B3} = \overline{\hat{\boldsymbol{y}}_{ik}^{(c_k)}}, \quad c_k \in \mathcal{C}^A(k), \quad k \in N \qquad [38]$$

This modeling approach provides a direct way of utilizing existing MMS models in the graph situation, which is called Benchmark 3. The parameter estimation steps are similar to those of the EDGM, as shown in Section 2 with the respective models [36] to [38]. The proposed EDGM is compared with the other three benchmarks in different scenarios by intensive simulation studies. Several important factors are considered, including (1) the number of nodes in the graph; (2) the density of the arcs in the graph; (3) the sample size of the training dataset; (4) the scale of transition error σ_z^2; and (5) the scale of emission error σ_ϵ^2. The detailed settings of these factors can be found in Table 1.

The benchmark combination of factor settings is G1 with 100 training samples and $\sigma_\epsilon^2 = \sigma_z^2 = 0.1$, for

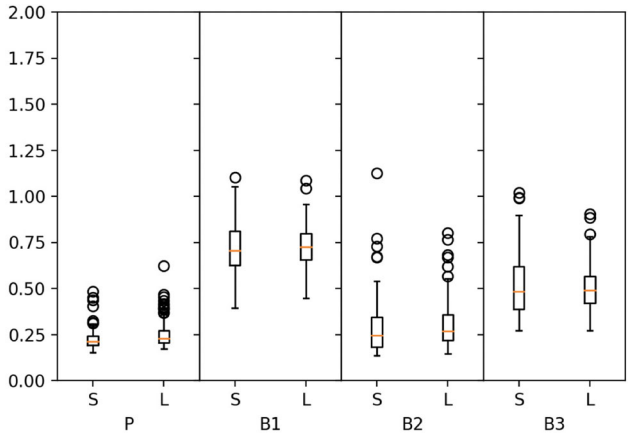


Figure 4. Boxplots of RMSEs from the proposed EDGM and three benchmarks with different numbers of nodes in graph (P: the proposed EDGM; B1–B3: Benchmarks 1–3; S: small number of nodes; L: large number of nodes).

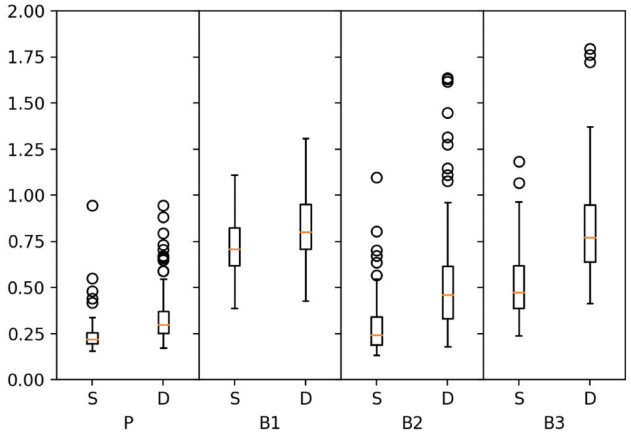


Figure 5. Boxplots of RMSEs from the proposed EDGM and three benchmarks with different densities of arcs in graph (P: the proposed EDGM; B1–B3: Benchmarks 1–3; S: sparse arcs; D: dense arcs).

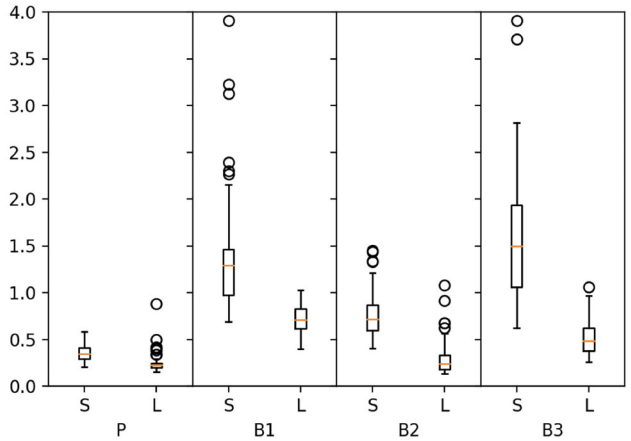


Figure 6. Boxplots of RMSEs from the proposed EDGM and three benchmarks with different training sample sizes (P: the proposed EDGM; B1–B3: Benchmarks 1–3; S: small training set; L: large training set).

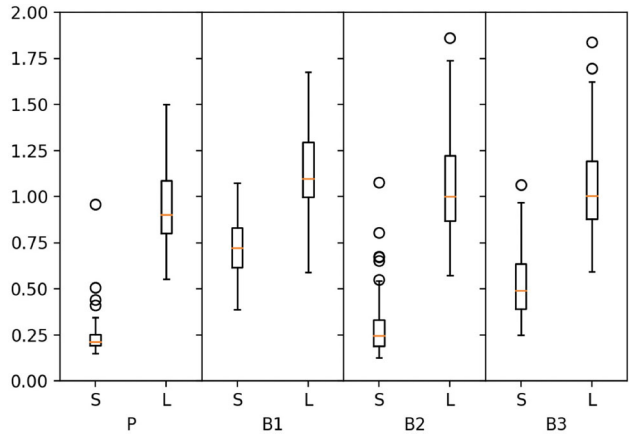


Figure 7. Boxplots of RMSEs from the proposed EDGM and three benchmarks with different scales of transition error (P: the proposed EDGM; B1–B3: Benchmarks 1–3; S: small scale; L: large scale).

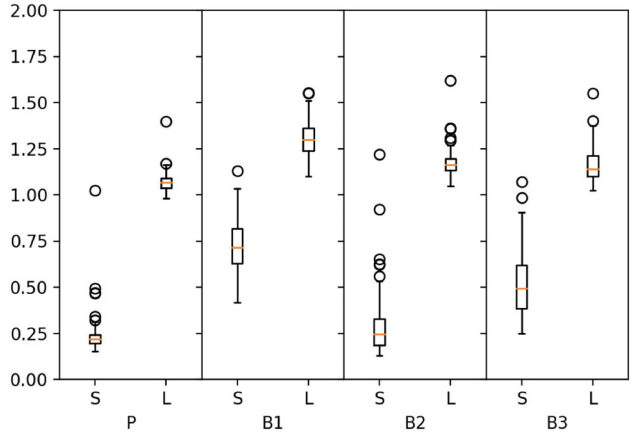


Figure 8. Boxplots of RMSEs from the proposed EDGM and three benchmarks with different scales of emission error (P: the proposed EDGM; B1–B3: Benchmarks 1–3; S: small scale; L: large scale).

example, when analyzing the effect of the number of nodes, the results from the proposed EDGM and three benchmarks in G1, G2, and G3 are compared with training sample sizes $N_{train} = 100$ and $\sigma_\epsilon^2 = \sigma_z^2 = 0.1$, and the results from G1 and G2 are compared when analyzing the effect of the density of the arcs in graph structures. Other settings for this part of the numerical experiment are given as the size of the test dataset is $N_{test} = 20$; the size of the validation dataset is set according to the size of the training dataset, that is, $N_{valid} = 0.5N_{train}$; and all the results for prediction accuracy are calculated under 300 replications. The validation metrics for choosing the combinations of hyperparameter Λ and evaluation of the prediction accuracy are the root mean square error (RMSE) of the validation dataset and test dataset, respectively, which are given as follows:

$$RMSE_{valid}(\Lambda) = \frac{1}{N_{valid}} \sum_{i=1}^{N_{valid}} \frac{1}{|\mathcal{N}|} \sum_{k \in \mathcal{N}} \frac{1}{n_{yk}} \| \boldsymbol{y}_{ik} - (\hat{\boldsymbol{y}}_{ik} | \boldsymbol{x}_{ij}, j \in \mathcal{N}, \Lambda) \|^2 \quad [39]$$

$$RMSE = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} \frac{1}{|\mathcal{N}|} \sum_{k \in \mathcal{N}} \frac{1}{n_{yk}} \| \boldsymbol{y}_{ik} - (\hat{\boldsymbol{y}}_{ik} | \boldsymbol{x}_{ij}, j \in \mathcal{N}, \Lambda_{best}) \|^2 \quad [40]$$

where Λ_{best} is the best combination of hyperparameters chosen by Eq. [39] and N_{valid} and N_{test} are the sizes of the validation dataset and test dataset, respectively. In Figures 4–8, a comparison of the proposed EDGM and benchmark models is shown by varying the five factors in Table 1. According to the results, several findings can be concluded: (1) the proposed EDGM outperforms the three benchmarks in terms of the means and standard deviations in RMSEs in nearly all scenarios, especially Benchmark 1 and Benchmark 3; (2) compared with the number of nodes, the prediction accuracy is more sensitive to the edge density. All these methods have larger means and standard deviations of RMSEs when the graph has denser arcs; (3) larger scales of transition error σ_z^2 result in larger means and standard deviations of RMSEs; (4) the proposed EDGM can obtain a robust estimation compared to the three benchmarks even with a small training sample size; (5) larger scales of emission error σ_ϵ^2 only result in larger means of RMSEs but have no substantial influence on standard deviations but a relatively smaller deviation of RMSEs. As shown in Figure 8, the RMSEs are not symmetrically distributed and are left truncated, where the threshold is determined by the model error. In this case, the threshold is mostly determined by the scale of emission error. On the other hand, the RMSEs' distributions have right tails caused by what we call "failure estimations." The reasons for failure estimations consist of two parts: (1) the overfitting problem since there are a large number of parameters to be estimated. This happens mostly in the EDGM because there are more estimated parameters in the EDGM than benchmarks which is why RMSEs for the EDGM have more outliers; (2) for the benchmarks, there exists systematic error since the simulated data are generated as the EDGM. The error caused by these two problems would slightly increase with the increase of emission error, while the left threshold is significantly influenced by the increase of emission error. Therefore, this would "compress" the distribution of RMSEs of the four methods, resulting in the finding that high emission error leads to smaller standard deviations of RMSEs.

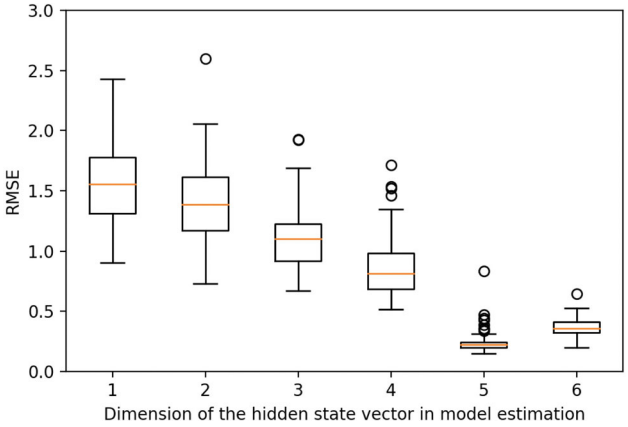


Figure 9. Boxplots of RMSEs from the proposed EDGM with different dimensions of the hidden state vector in estimating the model, that is, $\tilde{n}_h$ (the true dimension of the hidden state vector is $n_h = 5$).

Note that these comparisons are based on simulated data generated by EDGM Eqs. [5] and [6]. Although the advantages of the EDGM shown in Figures 4–8 might be explained by how the data are generated, the results still prove that the EDGM can better model the system if it has hidden states and graph structures of subsystems. On the other hand, using these benchmarks for model simplification cannot obtain a satisfactory result for predicting the outputs.

Additionally, we want to find the EDGM's performance when we don't know the true dimension of the hidden state vector. Denote the true dimension of the hidden state vector as n_h, that is, the dimension of the hidden state vector in generating the simulated data. In practice, we sometimes don't know n_h. Denote the dimension of the hidden state vector in formulating and estimating the model to be $\tilde{n}_h$. The figure below shows the EDGM's prediction performance with different $\tilde{n}_h$. n_h is set to be $n_h = 5$, and the other settings are the same as the benchmark setting. As shown in Figure 9, when $\tilde{n}_h \neq n_h$, the estimated EDGM's has both larger means and deviations of RMSEs. Specifically, when $\tilde{n}_h < n_h$, this difference is relatively large and would decrease as $\tilde{n}_h$ being closer to n_h. On the other hand, when $\tilde{n}_h > n_h$, this difference is smaller but increases the computation cost and possibility of overfitting because of the increasing number of estimated parameters. In practice, based on the dimension reduction technique, such as principal component analysis (PCA), if the size of the training dataset is large and with great computation ability, a larger dimension $\tilde{n}_h$ can be chosen to get better predictive performance; if the size of the training dataset is small with a high requirement on computation

speed, the chosen dimension of the hidden state vector $\tilde{n}_h$ can be set smaller.

In the second part, monitoring the effectiveness of the proposed GB-MEWMA control chart is verified. MEWMA and group EWMA are the two comparing control charts that are given in Eqs. [41] and [42] as follows:

$$MEWMA:\ \boldsymbol{z}_t^{\boldsymbol{y}} = (1-\lambda)\boldsymbol{z}_{t-1}^{\boldsymbol{y}} + \lambda \boldsymbol{y}_t,\quad t = 1, \ldots, \tau, \tau+1, \ldots, m$$

$$MZ_t^{MEWMA} = \sqrt{\frac{\lambda}{2-\lambda}}\left|\boldsymbol{z}_t^{yT}\boldsymbol{\Sigma}_z^{-1}\boldsymbol{z}_t^{\boldsymbol{y}}\right|,\ \boldsymbol{\Sigma}_z = \frac{\lambda}{(2-\lambda)}\boldsymbol{\Sigma}_y \quad [41]$$

$$GEWMA:\ \boldsymbol{z}_t^{\boldsymbol{k}} = (1-\lambda)\boldsymbol{z}_{t-1}^{\boldsymbol{k}} + \lambda \boldsymbol{e}_{tk},\ k \in \mathcal{N},\quad t = 1, \ldots, \tau, \tau+1, \ldots, m$$

$$MZ_t^{GEWMA} = \max_{k\in\mathcal{N}}\left(\sqrt{\frac{\lambda}{2-\lambda}}\left|\boldsymbol{z}_t^{\boldsymbol{k}T}\boldsymbol{\Sigma}_{z_k}^{-1}\boldsymbol{z}_t^{\boldsymbol{k}}\right|\right),\ \boldsymbol{\Sigma}_{z_k} = \frac{\lambda}{(2-\lambda)}\boldsymbol{\Sigma}_{e_k} \quad [42]$$

where MZ_t^{MEWMA} and MZ_t^{GEWMA} are the respective charting statistics; $\boldsymbol{e}_{tk}$ is the error vector of node k at the t th observation and $\boldsymbol{\Sigma}_{e_k}$ is the respective covariance matrix, which can be calculated according to $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ by Eqs. [20] and [21]. Details for the designs of these two control charts can be found in previous works (Alipour and Noorossana 2010; Xiang and Tsung 2008). Under graph structure G1 with $\sigma_z^2 = \sigma_\epsilon^2 = 1$, the monitoring ability of the proposed GB-MEWMA and the two benchmark control charts are compared in different scenarios varying in the scale of shifts δ and OC position $k^{OC} \in \mathcal{N}$. Detailed settings of these two factors are listed in Table 2.

Similar to Section 2.4, the OC models assume that a mean shift $\boldsymbol{s}_k = \delta 1_k$ occurs in the hidden state vector at node $k = k^{OC}$, as in Eq. [18]. With the specified IC ARL, $ARL_0 = 370$ and weight parameter $\lambda = 0.2$, the control limits of the three control charts are set. The means and standard deviations of the ARLs of different OC scenarios are shown in Table 3 under 2000 replications. We also consider combining

Table 2. Factor settings of the monitoring studies.

Factor	Scenarios
OC position k^{OC}	$k^{OC} \in \{2, 4, 6, 8\}$
The scale of shifts δ	$\delta \in \{0.5,\ 1,\ 1.5,\ 2,\ 3,\ 4,\ 5\}$

Table 3. ARL comparisons of different OC scenarios of GB-MEWMA and the other comparing control charts.

Delta	GB-MEWMA	MEMWA	GEWMA	B1 + GEWMA	B2 + GEWMA	B3 + GEWMA
			Shift at Node 2			
0.5	**160.69 (32.75)**	280.92 (28.96)	265.14 (25.15)	173.41 (20.44)	217.01 (23.28)	205.11 (26.32)
1	**81.50 (24.76)**	145.01 (19.05)	133.63 (17.82)	84.08 (9.08)	92.86 (13.34)	107.83 (12.68)
1.5	**36.14 (16.03)**	73.98 (13.10)	68.17 (12.85)	42.27 (6.68)	51.21 (6.76)	66.94 (7.14)
2	**22.9 (13.27)**	44.41 (9.55)	35.71 (8.29)	30.03 (2.04)	33.51 (9.20)	35.55 (9.99)
3	**12.32 (5.84)**	15.20 (4.68)	13.26 (4.48)	13.00 (1.46)	15.19 (3.56)	16.30 (4.77)
4	**5.76 (2.72)**	8.75 (3.38)	6.20 (2.60)	5.81 (0.82)	7.55 (1.39)	7.65 (2.36)
5	**3.46 (2.83)**	5.65 (3.21)	4.17 (2.34)	3.76 (0.64)	4.39 (1.33)	4.75 (1.74)
			Shift at Node 4			
0.5	**182.71 (34.62)**	280.65 (28.29)	265.22 (25.56)	205.16 (22.60)	236.06 (27.11)	231.72 (28.26)
1	**89.14 (26.06)**	145.22 (19.01)	123.03 (16.18)	94.70 (10.24)	108.81 (8.11)	112.85 (14.43)
1.5	**51.37 (15.67)**	74.50 (12.86)	68.53 (12.59)	59.60 (6.01)	60.55 (7.04)	67.03 (7.74)
2	**29.69 (10.51)**	46.76 (10.99)	32.11 (6.80)	30.87 (5.50)	30.63 (5.67)	31.71 (6.78)
3	**12.45 (5.81)**	16.40 (5.22)	12.52 (4.31)	13.49 (1.03)	12.92 (1.63)	15.37 (2.78)
4	**4.79 (2.82)**	8.26 (3.01)	5.31 (1.62)	5.65 (0.83)	5.40 (0.67)	5.77 (1.88)
5	3.22 (1.65)	4.05 (1.77)	4.01 (1.35)	**3.16 (0.82)**	3.26 (0.60)	3.68 (0.92)
			Shift at Node 6			
0.5	**277.21 (30.39)**	333.29 (29.60)	319.46 (28.48)	289.63 (27.44)	298.80 (35.74)	296.85 (27.77)
1	**175.30 (24.67)**	236.09 (26.35)	196.77 (21.96)	180.26 (20.59)	187.13 (22.17)	181.43 (23.97)
1.5	**97.32 (20.06)**	162.58 (21.94)	114.71 (17.03)	101.45 (18.20)	107.06 (15.98)	107.74 (16.74)
2	**69.54 (17.77)**	107.15 (17.43)	77.26 (14.80)	71.58 (11.13)	76.91 (12.49)	78.08 (12.71)
3	**38.46 (11.30)**	61.25 (13.98)	43.03 (8.22)	38.88 (9.98)	39.71 (10.34)	40.17 (10.74)
4	23.73 (8.06)	34.20 (9.39)	25.29 (8.20)	**18.85 (7.71)**	19.23 (7.73)	26.80 (7.75)
5	13.90 (6.73)	25.01 (8.99)	15.44 (6.10)	**8.91 (2.52)**	12.93 (2.90)	13.67 (2.74)
			Shift at Node 8			
0.5	**330.89 (31.95)**	354.43 (29.81)	333.38 (28.27)	335.59 (28.66)	341.54 (30.57)	351.13 (30.60)
1	**231.22 (28.64)**	299.50 (26.84)	260.59 (26.22)	235.22 (27.48)	242.13 (29.96)	251.73 (27.69)
1.5	**166.12 (25.54)**	223.50 (23.79)	179.44 (22.34)	169.25 (23.65)	174.17 (24.68)	169.14 (25.06)
2	**119.80 (22.88)**	181.22 (21.89)	135.64 (19.45)	126.05 (21.09)	131.96 (22.56)	140.71 (21.72)
3	82.18 (19.38)	121.28 (20.32)	89.50 (16.96)	**75.35 (19.32)**	80.41 (19.28)	82.29 (16.16)
4	65.48 (16.84)	90.07 (17.64)	69.12 (15.45)	**64.93 (17.05)**	65.76 (15.43)	68.61 (15.08)
5	49.02 (13.79)	68.06 (15.96)	52.97 (14.91)	**45.63 (14.38)**	46.91 (10.77)	48.51 (11.85)

Note: Bold values represent the best of each row.

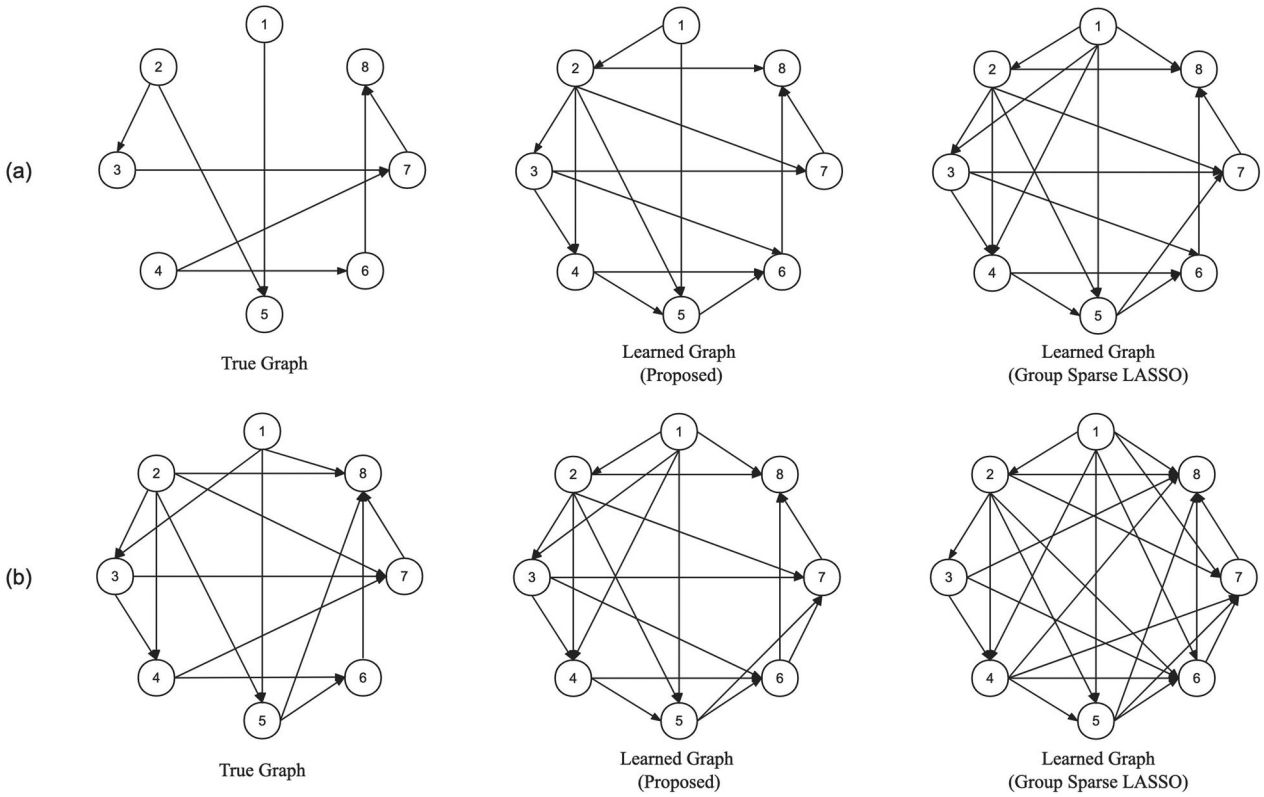


Figure 10. Comparisons of the true graph and learned graphs by the proposed adjusted MCP approach and group sparse LASSO: (a) G1, (b) G2.

modeling methods and monitoring methods as a whole framework. Among the three monitoring methods, only GEWMA is based on the prediction error, while the monitoring statistics of GB-MEWMA and MEWMA are calculated directly from the observed output variables. Note that the modeling methods only impact the prediction errors, which means the monitoring ability of GB-MEWMA and MEWMA is irrelevant to the modeling methods. Therefore, we have only added combinations of GEWMA and the three benchmarks, named as B1 + GEWMA, B2 + GEWMA, and B3 + GEWMA, respectively. As shown in the comparison results, GB-MEWMA outperforms the other two control charts in terms of the means of ARLs, especially MEWMA. The advantages of GB-MEWMA are larger with smaller scales of shifts and OC nodes that have more child nodes, which means that the proposed control chart is more effective when detecting small shifts and succeeds in characterizing the effect on other nodes through the graph structure. Although GB-MEWMA has larger standard deviations, it still remains in an acceptable range and decreases rapidly as shift scales increase. On the other hand, when combined with the estimated models, the advantages of the GB-MEWMA decrease as the OC node being away from the root node and the OC scale being larger. This finding further proves that the GB-MEWMA is more useful for detecting small shifts

Table 4. Confusion matrices and F_1 scores for G1 and G2 of the proposed adjusted MCP approach.

		Predicted graph	
G1		True	False
True graph	True	7	1
	False	7	13
F_1 score		0.746	
G2		Predicted graph	
		True	False
True graph	True	14	2
	False	6	6
F_1 score		0.636	

Table 5. Confusion matrices and F_1 scores for G1 and G2 of group sparse LASSO.

		Predicted graph	
G1		True	False
True graph	True	7	1
	False	11	9
F_1 score		0.594	
G2		Predicted graph	
		True	False
True graph	True	15	1
	False	9	3
F_1 score		0.395	

because of its ability to accumulate them through the graph structure.

In the third part, the proposed structure learning framework introduced is verified in structures of G1 and G2. As mentioned in Section 2.5, the given information of graph structures is turned from the parent node set $Pa(k)$ to the candidate parent node set $Cpa(k)$, where the settings for $Cpa(k)$ are usually learned from prior expert knowledge in real cases. In simulation studies, we assume there is no prior knowledge and set $Cpa(k) = \{j < k | j \in \mathcal{N} = \{1, ..., K\}\}$, which has no limits on the parents except for keeping the DAG assumption. The comparison method is group sparse LASSO, which learns parents separately by each node k and sets parameters from $\boldsymbol{U}_{jk}$ to be in the same group (Li, Nan, and Zhu 2015). The factor settings of the two graphs are the same as the benchmark setting in the parameter estimation part, that is, $\sigma_z^2 = \sigma_\epsilon^2 = 0.1$ and $N_{train} = 100$. Specially, in the numerical experiment part, since the size of training and validation dataset is not enough to select an optimal combination of the tuning parameters and to avoid overfitting problem on tuning parameters, we set $\lambda_1 = \lambda_2 = \lambda_{\mathcal{S}}$, and $\Lambda_1 = \Lambda_2 = \Lambda_{\mathcal{S}}$. Comparisons of the learned graph structures and confusion matrices for determining the existence of all possible arcs are shown in Figure 10 and Tables 4 and 5. The results show that the group sparse LASSO tends to learn a denser graph than the adjusted MCP approach in learning the EDGM. A reasonable explanation for this finding is that group sparse LASSO might regard some of the ancestor nodes that may not directly influence the aimed node as the parent nodes, while the proposed adjusted MCP approach can accurately predict the outputs with limited parents through the transition of hidden states under the EDGM and the framework of stochastic proximal gradient algorithm. In real cases, the proposed method for learning graph structures is also more attractive because a sparser structure provides more specific information about the system, even with a few missed arcs.

4. Case study

In this section, a monocrystalline silicon growth process using the CZ method is introduced to show how the proposed EDGM is applied in real-world cases to help model and predict essential output variables. As mentioned in Section 1, the CZ method is a mainstream technique for monocrystalline growth processes due to its low cost and efficiency compared with other methods (Mohamed Ariff, Hashmi, and Brabazon 2018). However, the growth process of a normal-sized silicon ingot usually takes a long time (approximately several days), and there would be much higher extra costs of materials and time to terminate the growth process before a whole ingot is produced. On the other hand, there are many important factors related to the CZ silicon growth process, such as thermal effects, impurities, and rotational speed of the seed. Small deviations in these factors might affect the final quality of the ingots, resulting in various defects, such as stress, point defects, and dislocations (Seigneur et al. 2016). Therefore, to better monitor and control the silicon growth process, numerous sensors are installed on the equipment. The variables monitored by these sensors, together with a great number of controllable variables, form the total variable set.

For the case studied in this article, all the involved variables can be further grouped into six subsystems: gas control, pressure field, heater, thermal field, machinery, and ingot. The available data contain ~4000 observations from 42 ingots, which are divided into a training set, a validation set, and a test set in the following proportions: $N_{train} : N_{valid} : N_{test} = 3 : 1 : 1$. The observations are collected from two important stages of the growth process: (1) the shoulder growth (SG) stage when the seed decreases its rotation and pulling speed to increase the ingot diameter and (2) the isodiametric growth (IG) stage when the seed keeps a steady speed to form a cylindrically shaped ingot with the required diameter. The total observations are separated by the two stages, where the SG stage takes a relatively shorter time (~300 observations) and the IG stage tends to be longer but steadier (~3700 observations). There are 30 controllable variables and 20 sensing variables, which are comparatively average-distributed in the six subsystems. The

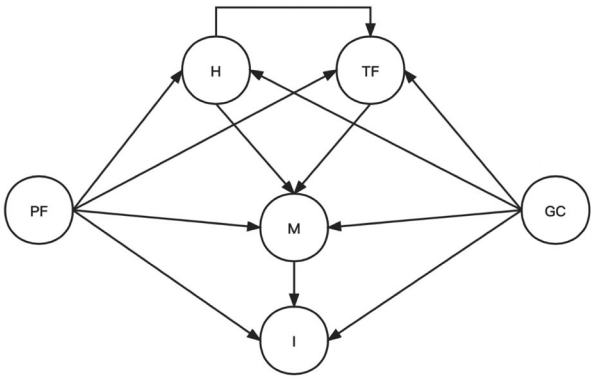
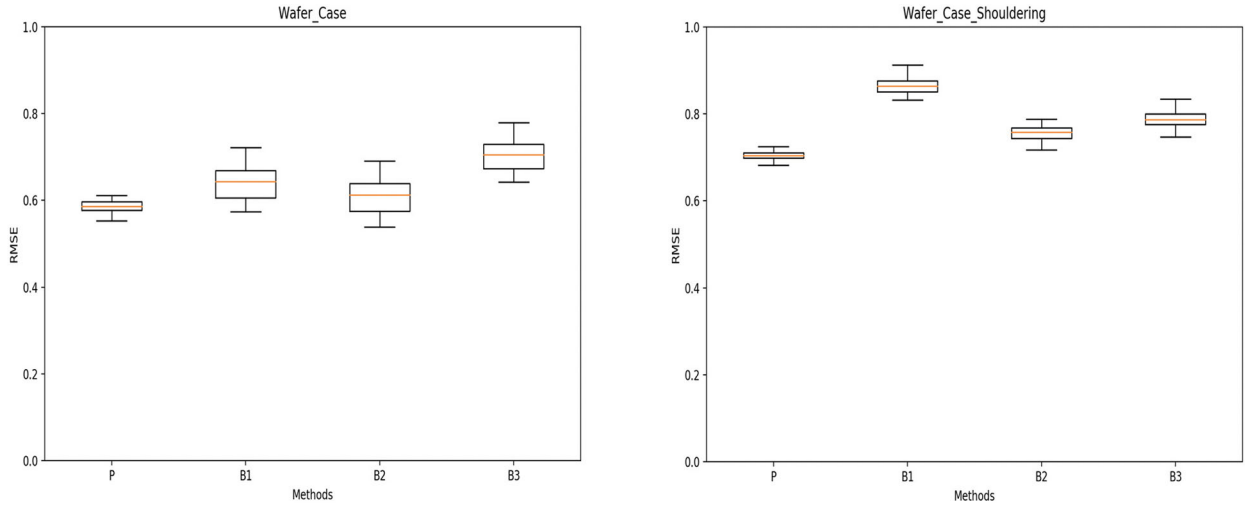


Figure 11. Graph structure of CZ silicon growth process in the SG and IG stages (PF: pressure field; GC: gas control; H: heater; TF: thermal field; M: machinery; I: ingot).



(a) IG stage (b) SG stage

Figure 12. Boxplots of RMSEs from the proposed EDGM and three benchmarks in two stages (P: the proposed EDGM, B1–B3: Benchmarks 1–3).

Table 6. RMSEs from the proposed EDGM and three benchmarks in two stages (standard deviations are shown in parentheses).

Isodiametric growth stage				
Methods	Proposed	Benchmark 1	Benchmark 2	Benchmark 3
RMSE	0.58478 (0.0139)	0.64143 (0.0388)	0.60956 (0.0407)	0.70443 (0.0351)
Shoulder growth stage				
Methods	Proposed	Benchmark 1	Benchmark 2	Benchmark 3
RMSE	0.70504 (0.0102)	0.86458 (0.0168)	0.75588 (0.0148)	0.78930 (0.0176)

dimensions of hidden state vectors are set to $n_h = 2$ in this case. Detailed information for variables is not introduced due to secrecy concerns.

Although controllable variables can be adjusted through a feedback control system during the growth process, they are usually set to predetermined values to avoid variations from the environment. Therefore, we need to model the relationship between variables and make accurate predictions for sensing variables, which contain important quality measurements given controllable variables. In this case, controllable variables and sensing variables are considered as inputs and outputs, respectively, while the subsystems are set as nodes in graph structures. Moreover, based on the CZ technique and the real situation, prior expert information on the graph structure is given in Figure 11: the heater subsystem influences the machinery subsystem through the thermal field; the machinery controls the rotation and pulling speed of the seed to affect the diameter of the ingot; and the pressure field and gas control subsystems determine the external environment to affect the whole growth process. Based on the known structure, the results of prediction accuracy for all output variables from the EDGM and the three benchmark models (the same as Section 3) are shown in Figure 12 and Table 6. We observe that the proposed EDGM has better prediction accuracy and stability in both stages. Compared with the IG stage, the SG stage has more changing variables in the outputs, resulting in smaller standard deviations but larger means of RMSEs in all methods. On the other hand, the goal of the IG stage is to keep the ingot growing at a required diameter; therefore, many outputs are comparatively steady, resulting in smaller means of RMSEs, while the overfitting problem results in a larger standard deviation in the three benchmark models. These observations prove that the proposed EDGM can better model the relationship between controllable variables and sensing variables in this silicon growth process by introducing hidden state vectors to select important factors that influence the outputs and other subsystems.

5. Conclusions

In this article, an extended directed graphical model together with parameter estimation, monitoring, and structure learning methods are proposed. This

proposed model is an extension of traditional graphical models that considers variables as nodes. To address the problem that considerable variables in complex systems might have certain structures for separation with unequal importance, variables are classified in advance, and a group of variables is considered to be a node. The EDGM models the relationship among variables within and between nodes by introducing hidden state vectors. In numerical simulations, first, the prediction RMSEs of the EDGM are compared with those of the other three models to show the effectiveness and accuracy of the proposed parameter estimation step. Second, the monitoring performance of the proposed GB-MEWMA is proven to outperform MEWMA and GEWMA in detecting mean shifts in graphical models. An adjusted MCP approach for learning graph structures is also proposed and proven to obtain a sparser estimation for graphs compared to group sparse LASSO. Finally, the proposed EDGM is verified in a case regarding the monocrystalline silicon growth process. The results imply that with hidden state vectors, the EDGM outperforms the other methods in cases where there are numerous variables with prior knowledge of separations for subsystems. For future research, adjustment to the proposed EDGM is needed to deal with high-dimensional data. Although the input dimension can be set at a large number, temporal and spatial information is not considered in the EDGM, which is important in dealing with high-dimensional data, such as functional data, images, and videos. Also, the effectiveness of the model in addressing problems with nonlinear relationships and multilayer hidden states still remains to be studied.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work was funded by the Key Program of National Science Foundation of China under Grant No. 71932006.

About the authors

Dengyu Li is currently a PhD student at Department of Industrial Engineering, Tsinghua University. He received his B.S. degree in Department of Industrial Engineering, Tsinghua University in 2020. His email is lidy20@mails.tsinghua.edu.cn. His research interests lie in data-driven methods in applications of quality prediction and control.

Kaibo Wang is a professor in the Department of Industrial Engineering, Tsinghua University, Beijing, China. He received his BS and MS degrees in Mechatronics from Xi'an Jiaotong University, Xi'an, China, and his PhD in Industrial Engineering and Engineering Management from the Hong Kong University of Science and Technology, Hong Kong. His research focuses on statistical quality control and data-driven system modeling, monitoring, diagnosis, and control, with a special emphasis on the integration of engineering knowledge and statistical theories for solving problems from the real industry.

ORCID

Kaibo Wang http://orcid.org/0000-0001-9888-4323

Data availability statement

The data that support the findings of this study are available from the corresponding author, upon reasonable request.

References

Alipour, H., and R. Noorossana. 2010. Fuzzy multivariate exponentially weighted moving average control chart. *The International Journal of Advanced Manufacturing Technology* 48 (9–12):1001–7. doi: 10.1007/s00170-009-2365-4.

Breheny, P., and J. Huang. 2011. Coordinate descent algorithm for nonconvex penalized regression, with application to biological feature selection. *The Annals of Applied Statistics* 5 (1):232–53. doi: 10.1214/10-AOAS388.

Bukkapatnam, S., M. Malshe, P. M. Agrawal, L. M. Raff, and R. Komanduri. 2006. Parametrization of interatomic potential functions using a genetic algorithm accelerated with a neural network. *Physical Review B* 74 (22): 224102. doi: 10.1103/PhysRevB.74.224102.

de Campos, L. 1998. Independency relationships and learning algorithms for singly connected networks. *Journal of Experimental & Theoretical Artificial Intelligence* 10 (4): 511–49. doi: 10.1080/095281398146743.

Deng, X., and R. Jin. 2015. QQ models: Joint modeling for quantitative and qualitative quality responses in manufacturing systems. *Technometrics* 57 (3):320–31. doi: 10.1080/00401706.2015.1029079.

Ding, Y., D. Ceglarek, and J. Shi. 2002. Fault diagnosis of multistage manufacturing processes by using state space approach. *Journal of Manufacturing Science and Engineering* 124 (2):313–22. doi: 10.1115/1.1445155.

Drouiche, N., P. Cuellar, F. Kerkar, S. Medjahed, T. Ouslimane, and M. Ould Hamou. 2015. Hidden values in kerf slurry waste recovery of high purity silicon. *Renewable and Sustainable Energy Reviews* 52:393–9. doi: 10.1016/j.rser.2015.07.114.

Fan, J., and R. Li. 2001. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association* 96 (456):1348–60. doi: 10.1198/016214501753382273.

Gómez, A. M. E., K. Paynabar, and M. Pacella. 2021. Functional directed graphical models and applications in root-cause analysis and diagnosis. *Journal of Quality Technology* 53 (4):421–37. doi: 10.1080/00224065.2020.1805380.

Heckerman, D., D. Geiger, and D. Chickering. 1994. *Learning Bayesian networks: The combination of knowledge and statistical data*. Vol. 20. San Francisco, CA: Morgan Kaufmann.

Huang, Q., and J. Shi. 2004. Variation transmission analysis and diagnosis of multi-operational machining processes. *IIE Transactions* 36 (9):807–15. doi: 10.1080/07408170490472999.

Li, Y., B. Nan, and J. Zhu. 2015. Multivariate sparse group lasso for the multivariate multiple linear regression with an arbitrary group structure. *Biometrics* 71 (2):354–63. doi: 10.1111/biom.12292.

Liu, Q., W. Zhu, and Z. Ma. 2022. *Multi-task support vector machine with generalized Huber loss*. Durham, NC: Research Square.

Mohamed Ariff, A. H., M. S. J. Hashmi, and D. Brabazon. 2018. Monocrystalline silicon grown using floating zone technique. In *Reference module in materials science and materials engineering*. Amsterdam: Elsevier.

Seigneur, H., N. Mohajeri, R. P. Brooker, K. O. Davis, E. J. Schneller, N. G. Dhere, M. P. Rodgers, J. Wohlgemuth, N. S. Shiradkar, G. Scardera, et al. 2016. Manufacturing metrology for c-Si photovoltaic module reliability and durability, part I: Feedstock, crystallization and wafering. *Renewable and Sustainable Energy Reviews* 59:84–106. doi: 10.1016/j.rser.2015.12.343.

Shi, J. 2006. *Stream of variation modeling and analysis for multistage manufacturing processes*. Boca Raton, FL: Taylor & Francis. doi: 10.1201/9781420003901

Sun, H., X. Deng, K. Wang, and R. Jin. 2016. Logistic regression for crystal growth process modeling through hierarchical nonnegative garrote based variable selection. *IIE Transactions* 48 (8):787–96. doi: 10.1080/0740817X.2016.1167286.

Sun, H., S. Huang, and R. Jin. 2017. Functional graphical models for manufacturing process modeling. *IEEE Transactions on Automation Science and Engineering* 14 (4):1612–21. doi: 10.1109/TASE.2017.2693398.

Wang, C., X. Zhu, S. Zhou, and Y. Zhou. 2021. Bayesian learning of structures of ordered block graphical models with an application on multistage manufacturing processes. *IISE Transactions* 53 (7):770–86. doi: 10.1080/24725854.2020.1786196.

Xiang, L., and F. Tsung. 2008. Statistical monitoring of multistage processes based on engineering models. *IIE Transactions* 40 (10):957–70. doi: 10.1080/07408170701880845.

Yan, H., N. D. Sergin, W. A. Brenneman, S. J. Lange, and S. Ba. 2021. Deep multistage multi-task learning for quality prediction of multistage manufacturing systems. *Journal of Quality Technology* 53 (5):526–44. doi: 10.1080/00224065.2021.1903822.

Zhang, C.-H. 2010. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics* 38 (2):894–942. doi: 10.1214/09-AOS729.

Zhang, H., L. Zheng, X. Ma, B. Zhao, C. Wang, and F. Xu. 2011. Nucleation and bulk growth control for high efficiency silicon ingot casting. *Journal of Crystal Growth* 318 (1):283–7. doi: 10.1016/j.jcrysgro.2010.10.103.

Zou, C., and F. Tsung. 2008. Directional MEWMA schemes for multistage process monitoring and diagnosis. *Journal of Quality Technology* 40 (4):407–27. doi: 10.1080/00224065.2008.11917746.

Zou, H., and R. Li. 2008. One-step sparse estimates in nonconcave penalized likelihood models. *Annals of Statistics* 36 (4):1509–33. doi: 10.1214/009053607000000802.