



Article Pre-Print

The following article is a “pre-print” of an article accepted for publication in an Elsevier journal.

Planning and Scheduling of Steel Plates Production. Part I. Estimation of Production Times via Hybrid Bayesian Networks for Large Domain of Discrete Variables, *Comp Chem Eng.*, 79 113-134 (2015)

The pre-print is not the final version of the article. It is the unformatted version which was submitted for peer review, but does not contain any changes made as the result of reviewer feedback or any editorial changes. Therefore, there may be differences in substance between this version and the final version of record.

The final, official version of the article can be downloaded from the journal’s website via this DOI link when it becomes available (subscription or purchase may be required):

[doi:10.1016/j.compchemeng.2015.02.005](https://doi.org/10.1016/j.compchemeng.2015.02.005)

This post-print has been archived on the author’s personal website (macc.mcmaster.ca) in compliance with the National Sciences and Engineering Research Council ([INSERC](#)) [policy on open access](#) and in compliance with [Elsevier’s academic sharing policies](#).

This post-print is released with a [Creative Commons Attribution Non-Commercial No Derivatives License](#).

Date Archived: June 1, 2016

Planning and Scheduling of Steel Plates Production. Part I: Estimation of Production Times via Hybrid Bayesian Networks for Large Domain of Discrete Variables

Junichi Mori and Vladimir Mahalec*

Department of Chemical Engineering
McMaster University, Hamilton, Ontario, Canada L8S 4L7

Abstract

Knowledge of the production loads and production times is an essential ingredient for making successful production plans and schedules. In steel production, the production loads and the production times are impacted by many uncertainties, which necessitates their prediction via stochastic models. In order to avoid having separate prediction models for planning and for scheduling, it is helpful to develop a single prediction model that allows us to predict both production loads and production times. In this work, Bayesian network models are employed to predict the probability distributions of these variables. First, network structure is identified by maximizing the Bayesian scores that include the likelihood and model complexity. In order to handle large domain of discrete variables, a novel decision-tree structured conditional probability table based Bayesian inference algorithm is developed. We present results for real-world steel production data and show that the proposed models can accurately predict the probability distributions.

Keywords: Production load, production time, hybrid Bayesian network, structure

*Corresponding author. Email: mahalec@mcmaster.ca

learning, large domain variable, decision tree, maximum likelihood estimation, steel plate production

1 Introduction

Accurate estimation of the production loads and the total production times in manufacturing processes is crucial for optimal operations of real world industrial systems. In this paper, a production load represents the number of times that a product is processed in the corresponding process unit and a production time is defined as the length of time from production start to completion. Production planning and scheduling for short, medium and long term time horizons employ various optimization models and algorithms that require accurate knowledge of production loads and total production times from information at each of the processing steps. The approaches to predict the production loads and total production times can be either based on mechanistic model or can employ data-driven techniques. Model-based prediction methods may be applied only if accurate mechanistic models of the processes can be developed. First principal models require in-depth knowledge of the processes and still cannot take into consideration all uncertainties that exist in the processes. Therefore, mechanistic models may not work well for prediction of production loads and production times of the real-world industrial processes. On the other hand, data-driven approaches do not require in-depth process knowledge and some advanced techniques can deal with the process uncertainties.

The most straightforward approach is to use the classical statistical models (e.g. regression models) that estimate the values of new production loads and a production time from the past values in the historical process data. The relationships between the targeted variables and other relevant variables are used to compute the statistical model that can predict the production loads and production times. However, such models are too simple to predict the nonlinear behaviour and estimate the system uncertainties. An alternative

simple method is to compute the average of these values per each production group that has similar production properties and then utilize the average values of each production group as the prediction (Ashayeria *et al.*, 2006). In this case the prediction accuracy significantly depends on the rules that govern creation of production groups and it may be challenging to find the appropriate rules from process knowledge only. To overcome this limitation, supervised classification techniques such as artificial neural networks (ANN), support vector machine, Fisher discriminant analysis and K-nearest neighbours (KNN) may be useful to design the rules to make production groups from historical process data. However, even though we can accurately classify the historical process data into an appropriate number of production groups, these methods do not consider model uncertainties and cannot handle missing values and unobserved variables. In addition, we are typically forced to have multiple specific models tailored for specific purposes, e.g. for production planning or for scheduling, which causes model maintenance issues and lack of model consistency.

Bayesian network (BN) models offer advantages of having single prediction model for predicting the production loads and total process times in planning and scheduling. Bayesian networks are also called *directed graphical models* where the links of the graphs represent direct dependence among the variables and are described by arrows between links (Pearl, 1988; Bishop, 2006). Bayesian network models are popular for representing conditional independencies among random variables under system uncertainty. They are popular in the machine learning communities and have been applied to various fields including medical diagnostics, speech recognition, gene modeling, cancer classification, target tracking, sensor validation, and reliability analysis.

The most common representations of conditional probability distributions (CPDs) at each node in BNs are conditional probability tables (CPTs), which specify marginal probability distributions for each combination of values of its discrete parent nodes. Since the real industrial plant data often include discrete variables which have large discrete domains,

the number of parameters becomes too large to represent the relationships by the CPTs. In order to reduce the number of parameters, context-specific independence representations are useful to describe the CPTs (Boutilier *et al.*, 1996). Efficient inference algorithm that exploits context-specified independence (Poole and Zhang, 2003) and the learning methods for identification of parameters of context-specific independence (Friedman and Goldszmidt, 1996; Chickering *et al.*, 1997) have been developed. The restriction of these methods is that all discrete values must be already grouped at an appropriate level of domain size since learning structured CPTs is NP-hard. However, typically discrete process variables have large domains and the task of identifying a reasonable set of groups that distinguish well the values of discrete variables requires in-depth process knowledge. To overcome this limitation, *attribute – value hierarchies* (AVHs) that capture meaningful groupings of values in a particular domain are integrated with the tree-structured CPTs (DesJardins and Rathod, 2008). However, since some discrete process variables do not contain hierarchal structures and thus AVHs cannot capture the useful abstracts of values in that domain. In addition, this model cannot handle the continuous variables without discretizing them. Furthermore, the authors do not describe how to apply AVH-derived CPTs to Bayesian inference. Therefore, this method has difficulty predicting probability distributions of production loads and total process time from observed process variables in the real-world industrial processes. Efficient alternative inference methods in Bayesian Networks containing CPTs that are represented as decision trees have been developed (Sharma and Poole, 2003). The inference algorithm is based on variable elimination (VE) algorithm. However, because the computational complexity of the exact inference such as VE grows exponentially with the size of the network, this method may not be appropriate for Bayesian networks for large scale industrial data sets. In addition, the method does not deal with application of the decision-tree structured CPTs to hybrid Bayesian networks, where both discrete and continuous variables appear simultaneously.

In hybrid Bayesian networks, the most commonly used model that allows exact in-

ference is the conditional linear Gaussian (CLG) model (Lauritzen, 1992; Lauritzen and Jensen, 2001). However, the proposed network model does not allow discrete variables to have continuous parents. To overcome this limitation, the CPDs of these nodes are typically modeled as softmax function, but there is no exact inference algorithm. Although an approximate inference via Monte Carlo method has been proposed (Koller *et al.*, 1999), the convergence can be quite slow in Bayesian Networks with large domain of discrete variables. The other approach is to discretize all continuous variables in a network and treat them as if they are discrete (Kozlov and Koller, 1997). Nevertheless, it is typically impossible to discretize the continuous variables as finely as needed to obtain reasonable solutions and the discretization leads to a trade-off between accuracy of the approximation and cost of computation. As another alternative, the mixture of truncated exponential (MTE) model has been introduced to handle the hybrid Bayesian networks (Moral *et al.*, 2001; Rumi and Salmeron, 2007; Cobb *et al.*, 2007). MTE models approximate arbitrary probability distribution functions (PDFs) using exponential terms and allow implementation of inference in hybrid Bayesian networks. The main advantage of this method is that standard propagation algorithms can be used. However, since the number of regression coefficients in exponential functions linearly grows with the domain size of discrete variables, MTE model may not work well for the large Bayesian networks that are required to represent the industrial processes.

Production planning and scheduling in the steel industry are recognized as challenging problems. In particular, the steel plate production is one of the most complicated processes because steel plates are high-variety low-volume products manufactured on order and they are used in many different applications. Although there have been several studies on scheduling and planning problems in steel production, such as continuous casting (Tang *et al.*, 2000; Santos *et al.*, 2003), smelting process (I. Harjunkoski, 2001) and batch annealing (Moon and Hrymak, 1999), few studies have dealt with steel plate production scheduling. Steel rolling processes manufacture various size of plates from a wide range of

materials. Then, at the finishing and inspection processes, malfunctions occurred in the upstream processes (e.g. smelting processes) are repaired, and additional treatments such as heat treatment and primer coating are applied such that the plates satisfy the intended application needs and satisfy the demanded properties. In order to obtain successful plans and schedules for steel plate production, it is necessary to determine the production starting times that meet both the customer shipping deadlines and the production capacity. This requires prediction models that can accurately predict the production loads of finishing and inspection lines and the total process time. However, due to the complexity and uncertainties that exist in the steel production processes, it is difficult to build the precise prediction models. These difficulties have been discussed in the literature (Nishioka *et al.*, 2012).

In our work, in order to handle the complicated interaction among process variables and uncertainties, Bayesian networks are employed for predicting of the production loads and prediction of the total production time. Since the steel production data have large domain of discrete variables, their CPDs are described by tree structured CPTs. In order to compute the tree structured CPTs, we use decision trees algorithm (Breiman *et al.*, 1984) that is able to group the discrete values to capture important distinctions of continuous or discrete variables. If Bayesian networks include continuous parent nodes with a discrete child node, the corresponding continuous variables can be discretized as finely as needed, because the domain size of discretized variable does not increase the number of parameters in intermediate factors due to decision-tree structured CPTs. Since the classification algorithms are typically greedy ones, the computational task for learning the structured CPTs is not expensive. Then, the intermediate factors can be described compactly using a simple parametric representation called the canonical table representation.

As for Bayesian network structure, if the cause-effect relationship is clearly identified from process knowledge, knowledge based network identification approach is well-suited. However, the identification of cause-effect relationships may require in-depth knowledge of

the processes to characterize the complex physical, chemical and biological phenomena. In addition, it can be time-consuming to build precise graphical model for complex processes, and it is also challenging to verify the identified network structure. Therefore, data-driven techniques are useful to systematically identify the network structure from historical process data. The basic idea of learning the network structure is to search for the graph so that the likelihood based score function is maximized. Since this is a combinational optimization problem which is known as *NP*-hard, it is challenging to compute the precise network structure for industrial plants where there are a large number of process variables and historical data sets. In order to restrict the solution space, we fix the root and leaf nodes by considering the properties of the process variables. In addition, we take into account two types of Bayesian network structures, which are: (i) a graph where a node of the total production time is connected with all nodes of the production loads and (ii) a graph where a node of the total production time is connected with all nodes of the individual production time on each production load and the nodes of these individual process time are connected to the nodes of the total production time. In the latter Bayesian network structure, the training data set that contains each production period of all production processes is needed in order to directly compute the parameters of conditional probability distributions of each production period.

We also propose the decision-tree structured CPT based Bayesian inference algorithm, which employs belief propagation algorithm to handle hybrid Bayesian networks with large domain of discrete variables. In order to reduce the computational effort required for multiplying and marginalizing factors during belief propagation, the operations via dynamically construct structured CPTs are proposed. Since the inference task is to predict the cause of production loads and the total production time from the effect of other process variables, we need the causal reasoning which can be carried out by using the sum-product algorithm from top node factors to bottom node factors. In addition, other types of inference such as diagnostic and intercausal reasoning are also investigated in the case when we need to infer

some other process variables such as operating conditions given production loads.

The organization of this paper is as follows. Section 2 introduces the plate steel production problem. Section 3 proposes the structure identification algorithm for steel plate production processes. Section 4 describes the proposed decision-tree structured CPTs based Bayesian inference algorithm and section 5 proposes the inference algorithm. The presented method is applied to the steel plate production process data in section 6. Finally, the conclusions are presented in Section 7. Appendix Appendices A describes algorithm to find the network structure that maximizes the score function, Appendix B describes computation of causal reasoning, Appendix C deals with diagnostic and causal reasoning, and Appendix D describes belief propagation with evidence.

2 Problem Definition

The plate steel mills studied in this paper consist of rolling machines, finishing and inspection processes. The purpose of the plate mills is to manufacture the steel plates, which are end products made from steel slabs via continuous casting processes. The slabs are sent to the rolling machines which include roughing mill, finishing mill and hot levelling machine through continuous or batch reheating furnace. To manufacture the high-strength steel plates, Accelerated Cooling (ACC) process which is based on the combination of controlled rolling and controlled cooling is used. After rolling or cooling, the plates are sent to the finishing and inspection processes, where defects that have occurred in the upstream processes such as smelting processes are repaired. Here, additional treatments such as heat treatment and primer coating are carried out as needed so that the plates have the properties required for their intended usages. The process is illustrated in Fig. 1.

One of the most important goals for production planning and scheduling is to use in the best possible manner the subsystems which are the bottlenecks, since the total plant

capacity is determined by the bottleneck capacity. In the plate mills, bottlenecks typically occur in the finishing and inspection lines. Therefore, an effective way to approach the planning and scheduling in the plate mills is to determine the rolling starting time of each order in such a manner that the limiting capacities are used to their maximum and that the customer demanded shipment deadlines are satisfied. For this purpose, when sales department staff receive orders from the customers, they must take into account the plate mills capacities and production times to avoid creating supply bottlenecks and to meet the customer deadlines. Similarly, the staff in the manufacturing department prepare the production plans and schedules while considering the production capacities and production times. However, due to the complicated and stochastic processes of the finishing and inspection lines, there is no deterministic way to predict the production loads and production time that are required to determine the production start times as required to avoid the bottlenecks and meet the customer deadline. In this paper, production loads represent the number of times that one plate or a set of total plates per day is processed at the corresponding production process unit. For instance, if 10 plates are processed at process A per day, then the production load of process A is computed as 10 per day. Production time for each plate or each customer order represents the time required to manufacture a product, starting from slabs to the finished products. In this paper, since the processing time in the inspection and finishing lines is a large part of the production time, the production time represents the period from production start to completion as shown in Fig. 1.

When a steel plate manufacturer receives orders from a customer, the product demand is known while the operating conditions, which are required to manufacture this specific product (order), are not available. On the other hand, at the manufacturing stage, all information about customer demands and detailed operating conditions is known.

Motivated by the above considerations, we can divide the tasks in plate steel production

planning and scheduling into two parts: (i) accurate prediction of the production loads and production time and (ii) optimization of the manufacturing plans and schedules based on the production time prediction models. In this paper, we focus on the development of the prediction model for production loads and production time. The prediction model needs to enable:

1. Estimation of the probability distributions of production loads and production time, since the finishing and inspection lines include various sources of uncertainties and thus stochastic prediction models are desirable.
2. Dealing with unobservable (unavailable) variables, because it is desirable to have a single model and avoid multiple models that meet with specific problems.

Having a single model with above properties enables planning and scheduling based on the same model which improves consistency of decisions between planning and scheduling and also simplifies model maintenance.

3 Identification of Bayesian Networks Structure for Steel Plate Production

In order to build a model which has the properties mentioned in the previous section, we develop the methods to compute probability distributions of unknown states of production loads and production times by using Bayesian network. Methodology to build the model will be illustrated by an example which has 21 discrete variables and 5 continuous variables as shown in Table 1. Each production load is assigned a binary variable having value of 1 if the corresponding production process is not needed, otherwise it is 2.

In this section, we propose the knowledge based data-driven structure learning algorithm to construct the Bayesian network structure. In the subsequent sections, we will propose

the parameter estimation and inference approaches with the constructed Bayesian network structure.

The computational task for structure learning is to find the optimal graph such that the maximized Bayesian scores can be obtained, which is known as NP-hard problem. In order to reduce the computational effort, we allow some variables to be set at the root nodes or the leaf nodes and also impose some constraints about orders of subsets of system variables.

First and the most important task for Bayesian inference is to synthesize the precise network structure. The most straightforward method is to design the network structure from process knowledge. However, by-hand structure design requires in-depth process knowledge because identification of cause-effect relationships is needed to characterize the complex physical, chemical and biological phenomena in systems. On the other hand, data-driven based techniques such as score function based structure learning are useful when there is lack of process knowledge or considered processes are too complicated to analyze.

The goal of structure learning is to find a network structure \mathcal{G} that is a good estimator for the data \mathbf{x} . The most common approach is *score-based structure learning*, which defines the structure learning problem as an optimization problem (Koller and Friedman, 2009). A score function $\text{score}(\mathcal{G} : \mathcal{D})$ that measures how well the network model fits the observed data \mathcal{D} is defined. The computational task is to solve the combinational optimization problem of finding the network that has the highest score. The detailed algorithm to find the network structure that maximizes the score function is explained in Appendix A. It should be noted that this combinational optimization problem is known to be *NP-hard* (Chickering, 1996). Therefore, it is challenging to obtain the optimal graph for industrial plants which often include a large number of process variables.

In order to reduce the computational cost, we incorporate the fundamental process knowledge into the network structure learning framework. As shown in the variable list of Table

1, all of our system variables belong to either customer demands or operating conditions or production loads or production time. One can immediately notice that customer demands can be considered “*cause variables*” of all system variables in the Bayesian network and are never impacted by either operating conditions, production loads or production time. Therefore, the nodes of variables belonging to the customer demands should be placed on the root nodes in the Bayesian network. Similarly, production loads are affected by the customer demands, operating conditions or both and not vice versa. Taking into account above considerations, we propose two types of Bayesian networks as shown in Fig. 2. In both Bayesian networks, the nodes related to the customer demands are placed at the root nodes followed by operating conditions, production loads and production time. The difference between two structures is that in the second BN structure, the nodes associated with the production time at each production process are added between the nodes of production loads and total production time while in the first BN structure, total production time is directly connected to the production loads.

In addition, we propose some predetermined variable ordering \prec over x to reduce further the computational cost further. If the ordering covers order relationships over all variables as $x_1 \prec x_2, \dots, x_I$ and the maximum number of parents for each node set to be at most d , then the number of possible parent sets for x_i is at most $\binom{i-1}{d}$ rather than 2^{i-1} (Koller and Friedman, 2009). In practice, nevertheless, these complete orderings are very difficult to identify from process knowledge. Instead of identification of complete orderings, we proposed *incomplete orderings* that do not cover order relationships over all variables but restrict the ordering over subsets of variables. The following is an example of the incomplete ordering.

$$\{x_1, x_2\} \prec \{x_3, x_4, x_5\} \prec \{x_6, x_7\} \quad (1)$$

This incomplete ordering suggests that a computed network should be consistent with the order relationships where x_1 and x_2 precede x_3, x_4, x_5, x_6 and x_7 , but any constraints about orders within each subset $\{x_1, x_2\}$, $\{x_3, x_4, x_5\}$ and $\{x_6, x_7\}$ are not imposed. Although

the incomplete orderings are optional to set for the structure learning, they can further restrict the solution space because they make the combinational search space much smaller. Therefore, our optimization task is to search for the optimal edges between the nodes of customer demands, operating conditions and production loads so that the BIC scores are maximized and the computed network is consistent with the fixed root and leaf nodes and the incomplete orderings. Consequently, we can further reduce the computational cost due to these constraints.

The search space in our optimization is restricted to the neighbouring graphs that can be obtained by either adding one edge or deleting one edge or reversing the existing edge. It should be noted that the neighbouring graphs should satisfy the constraints including fixed root and leaf nodes and graph to be acyclic. Therefore, the illegal graphs that violate these constraints are removed from neighbouring solutions during search. Tabu search algorithm (Glover, 1986) is employed to make a decision whether the neighbouring graphs are accepted. This algorithm uses a neighbourhood search to move from one solution candidate to an improved solution in the search space unless the termination criterion is satisfied. In order to avoid local optima as much as possible, a solution candidate is allowed to move to a worse solution unless that solution is included in tabu list which is a set of solutions that have been visited over the last t iterations, where t is termed as tabu size (set to be 100 in this work). The detailed algorithm for network identification is presented in Table 2.

4 Parameter Estimation for Bayesian Network with Large Domain of Discrete Variables

The major issue that is encountered when applying the proposed Bayesian networks is that some discrete variables have large discrete domains, which causes less robust inference. Let us assume that we have the conditional probability distribution of the node of UST load

whose parent nodes are customer demand A, B, C, D, F, G, H, I, and operating condition A. Since the product of cardinalities of all variables is $12 \times 2 \times 20 \times 20 \times 10 \times 10 \times 10 \times 10 \times 5 = 480$ million, it requires a huge amount of training data to compute the appropriate parameters for all elements in the table.

In order to overcome this issue, we propose the decision tree structured CPDs based Bayesian inference techniques that can handle a large number of discrete values. As for second issue of unobservable production time of each process, we utilize the maximum log-likelihood strategies to estimate the parameters of the probability distributions of each process time, as discussed in section 6.

The network structures studied in this work include both continuous and discrete variable nodes, and hence are called hybrid Bayesian networks. We represent sets of random variables by $\mathcal{X} = \Gamma \cup \Delta$ where Γ denotes the continuous variables and Δ represents the discrete variables. Random discrete variables are denoted by upper case letters from the beginning of the alphabet (A, B, A_1) while continuous variables are denoted by upper case letters near the end of the alphabet (X, Y, X_1). Their actual values are represented by the lower case letters (e.g. $a, x, \text{etc.}$). We denote sets of variables by bold upper case letters (\mathbf{X}) and the assignments of those sets by the bold lower case letters (\mathbf{x}). We use $\text{Val}(X)$ to represent the set of values that a random variable X can take.

The conditional linear Gaussian (CLG) model is widely used for representation of hybrid Bayesian networks. Let $X \subseteq \Gamma$ be a continuous node, $A \in \text{pa}(X) \cap \Delta$ be its discrete parent nodes and $Y_1, \dots, Y_k \in \text{pa}(X) \cap \Gamma$ be its continuous parent nodes where $\text{pa}(X)$ denotes a state of parent nodes of X . A Gaussian distribution of X for every instantiation $a \in \text{Val}(A)$ can be represented in moment form as:

$$p(X|\text{pa}(X)) = P(X|\text{pa}(X); \Theta_a) = \mathcal{N}\left(X \left| \sum_j^k w_{a,j} y_j + b_a, \sigma_a^2 \right.\right) \quad (2)$$

where $w_{a,j}$ and b_a are parameters controlling the mean, σ_a is the standard deviation of the

conditional distribution for X , and $\Theta_a = \{w_{a,j}, b_a, \sigma_a\}$ is the set of model parameters for instantiation a . This representation can also be rewritten in more convenient canonical form as follows (Lauritzen and Wermuth, 1989):

$$C(\mathbf{X}; K_a, h_a, g_a) = \exp\left(-\frac{1}{2}\mathbf{X}^T K_a \mathbf{X} + \mathbf{h}_a^T \mathbf{X} + g_a\right) \quad (3)$$

where K_a , \mathbf{h}_a and g_a represent parameters of a -th instantiation in canonical table representations. The canonical table representation can express both the canonical form used in continuous networks and the table factors used in discrete networks. Specifically, when $A = \emptyset$, only a single canonical form ($a = 1$) is obtained. Meanwhile, when $X = \emptyset$, parameters of K_a and \mathbf{h}_a are vacuous and only canonical forms $\exp(g_a)$ remain for every instantiation a .

The limitation of the canonical table representation is that discrete child nodes cannot have continuous parents. In our Bayesian networks shown in Figs. 10 and 11, since the variables related to size of plates such as height, width, length and weight are continuous nodes with discrete child nodes, they are converted into discrete variables. It is true that there is a trade-off between the accuracy of the approximation and the domain size of discretized variables, but due to the proposed decision-tree structured CPTs, we can deal with a large number of values in discrete variables when they are the root nodes in the Bayesian network. In our Bayesian networks, although the variables of height, width, length and weight are continuous ones with discrete child nodes, they do not have the parent nodes and thus we can discretize them as finely as we like.

Before computing the parameters of the canonical table, we need to define the instantiations of the set of discrete variables. The simplest approach is to use the traditional CPTs. However, the number of parameters required to describe such CPTs exponentially increases with the domain size. To overcome this limitation, it is useful to capture conditional independence that holds only in certain contexts in the Bayesian networks. Although several works about structured CPTs have been reported to capture this independence (Boutilier

et al., 1996; Poole and Zhang, 2003), all of these methods assume that all discrete variables are already grouped at an appropriate level of domain size.

In this work, classification trees algorithm (Breiman *et al.*, 1984) is employed to learn the structured CPTs. Classification trees can predict the values of the target variables by following decisions in the tree from the root nodes down to the leaf nodes. Their structures are identified by choosing a split so that the minimized statistical measures such as Gini impurity or entropy can be obtained.

First, we consider a factor that contains both continuous nodes \mathbf{X} and discrete nodes \mathbf{D} with continuous child node Y as shown in Fig. 3. Continuous variables are described as rectangles while discrete variables are ovals. The classification tree \mathcal{T} classifies the discrete variable set \mathbf{D} into a small number of subsets $A = \{a_1, a_2, \dots, a_L\}$ in accordance with the decision $a = \mathcal{T}(\mathbf{d})$ where L is the number of the leaves in the tree \mathcal{T} . For example, the decision tree provided in Fig. 4 means that if $D_1 \in \{4, 5, 6\}$, $D_3 \in \{4, 5\}$, then the corresponding records are assigned the instantiation a_5 regardless of the value of D_3 . It should be noted that the tree representation requires only $|\text{Val}(A)| = 6$ parameters to describe the behavior of target variable Y . In the example, if the traditional CPTs are used, the number of parameters becomes $|\text{Val}(D_1)| \times |\text{Val}(D_2)| \times |\text{Val}(D_3)| = 216$, which is much larger than that of the tree structured CPTs. Therefore, the tree structured CPTs need much smaller number of training data to learn the parameters of the conditional probability distributions and also lead to more robust estimation compared to the traditional CPT representation.

After giving instantiations $a_i \in A$ to all samples in the training data by following the decision in the constructed tree, parameters of the canonical for each instantiations s_i can

be computed as follows:

$$\begin{aligned} K_{a_i} &= \Sigma_i^{-1} \\ \mathbf{h}_{a_i} &= \Sigma_i^{-1} \mu_i \end{aligned} \tag{4}$$

$$g_{a_i} = -\frac{1}{2} \mu_i^T \Sigma_i^{-1} \mu_i - \log \left((2\pi)^{n/2} |\Sigma_i|^{1/2} \right)$$

with

$$\Sigma_i = \text{cov} [\mathbf{Z}_i] \tag{5}$$

$$\mu_i = \text{mean} [\mathbf{Z}_i] \tag{6}$$

where $\mathbf{Z} = \mathbf{X} \cup Y$, $\mathbf{Z}_i = \{z : \mathcal{T}(\mathbf{d}) = a_i\}$ and $\text{cov} [\mathbf{Z}_i]$ and $\text{mean} [\mathbf{Z}_i]$ are the covariance matrix and mean vector of \mathbf{Z}_i respectively.

Next, let us consider the case when a factor contains discrete nodes \mathbf{D} with discrete child node B as shown in Fig. 5. The classification tree classifies discrete variables \mathbf{D} into a small number of subsets $A = \{a_1, a_2, \dots, a_L\}$ by the following the rules $a = \mathcal{T}(\mathbf{d})$ with L being the number of leaves so that the values of a discrete child node B can be categorized well. While the decision tree can well categorize the discrete parent variables \mathbf{D} , a discrete child variable B that is used as a dependent variable in the tree \mathcal{T} is not included in the nodes of the decision tree. As a result, the obtained decision tree \mathcal{T} can classify values of discrete parent variables \mathbf{D} only and it excludes a discrete child variable B from the branching nodes in the tree \mathcal{T} . Since we need to classify values of all discrete variables that the factor includes, the discrete child node is placed at all leaf nodes in the tree \mathcal{T} to continue categorizing discrete variables by using their values. Specifically, all values of a discrete child variable $b_1 \dots b_L \in B$ are used as categorical splits at the leaf nodes in the tree \mathcal{T} and thus the number of branches at each leaf node is same as the domain size of $\text{Val}(B) = L$. Therefore, the total number of instantiations assigned in the new decision tree \mathcal{T}_{new} is $\text{Val}(B)$ times as large as the number of leaves in the original tree \mathcal{T} . Fig. 6 shows the example of a decision tree when a factor contains a discrete child node. The decision tree gives us the

rule where if $D_1 \in \{4, 5, 6\}$, $D_3 \in \{4, 5\}$ and $B \in \{1\}$, then the corresponding records are assigned the instantiation a_{4L+1} regardless of the value of D_2 . The number of instantiations are $\text{Val}(B) \times 6 = 6L$ rather than $|\text{Val}(D_1)| \times |\text{Val}(D_2)| \times |\text{Val}(D_3)| \times |\text{Val}(B)| = 216L$ which is the number of parameters required to describe the traditional CPT of discrete variables $\mathbf{D} \cup B$.

Using the instantiations $a_i \in A$ assigned by following the decision \mathcal{T}_{new} , parameters g_{a_i} of the canonical table representation can be computed as follows:

$$g_{a_i} = \log \frac{1}{N_{a_i}} \sum_z \mathbf{1}(f(\mathbf{d}) = a_i) \quad (7)$$

where $\mathbf{1}$ represents an indicator function, N_{a_i} is the number of samples belonging to the instantiation a_i . The other canonical parameters K_{a_i} and \mathbf{h}_{a_i} are vacuous since the factor does not include any continuous nodes.

5 Inference in Bayesian Network with Large Domain of Discrete Variables

5.1 Types of Reasoning

Bayesian inference answers the *conditional probability query*, $P(\mathcal{X} | \mathbf{E} = \mathbf{e})$ where the evidence $\mathbf{E} \subseteq \mathcal{X}$ is a subset of random variables in the Bayesian network and $\mathbf{e} \in \text{Val}(\mathbf{E})$ is the set of values given to these variables. The four major reasoning patterns in Bayesian inference are *causal reasoning*, *diagnostic reasoning*, *intercausal reasoning* and *mixed reasoning*. The causal reasoning predicts the downstream effects of factors from top nodes to bottom nodes while the diagnostic reasoning predicts the upstream causes of factors from bottom nodes to top nodes. The intercausal reasoning is the inference where different causes of the same effect can interact. The mixed inference is the combination of two or more of the above reasoning. In order to implement the causal inference, the conditional probability of

effects for given causes is computed by means of the sum-product algorithm from top node factors to bottom node factors.

Inferences other than causal reasoning, e.g. diagnostic, intercausal and mixed reasoning cannot be carried out in logic which is monotonic. Therefore, the fundamental algorithm for exact inference for these reasonings in graphical models is *variable elimination* (VE). VE can be carried out in a clique tree and each clique in the clique tree corresponds to the factors in VE. Since the computational cost of the clique tree algorithm exponentially grows with the width of the network, the inference cannot be achieved in large width networks that represent many real-world industrial problems. Hence, one needs to employ approximate inference algorithms such as loopy belief propagation (LBP) algorithm, since they are tractable for many real-world graphical models. Consequently, we will employ LBP algorithm to infer the probability distributions of unobserved variables, except for the causal reasoning. All of the above inference techniques require the computation of multiplying factors and marginalizing variables in factors.

Since our purpose for employing Bayesian inference is to estimate the downstream effects of production loads and production time from upstream causes of the customer demands or the operating conditions, we need to be able to carry out causal reasoning computations. We can carry out such Bayesian inference in the very straightforward way by applying the sum-product algorithm from top node factors to bottom node factors. However, it is also very useful to estimate the probability distributions of operating conditions given the customer demands and production loads in the case when we need to find the best operating conditions that satisfy both the customer demands and production capacities. Hence, besides the causal reasoning, in this work we take into account the other types of inference such as intercausal reasoning.

5.2 Multiplying Factors and Marginalizing Over Variables in Factors

In order to compute potentials of final and intermediate factors including messages between clusters, the operations of multiplying and marginalizing factors are needed. These operations require us to divide the large domain of discrete variables by following the tree structure. In this section, we develop two main operations: (i) multiplying factors and (ii) marginalizing over variables in factors.

The simplest approach is to convert the tree structured CPTs into traditional conditional probability tables which can be analyzed by using standard multiplying and marginalizing techniques. However, the number of rows in the converted traditional CPT of a factor exponentially increases both with the domain size and with the number of discrete variables that a factor includes. Therefore, this approach easily runs out of available computer memory for large Bayesian networks where factors contain a large number of discrete variables (huge domain of discrete variables). Instead of converting to the traditional CPTs, we propose more compact form of conditional probability tables.

Let us consider the simple example of multiplying two factors described in Fig. 7. First we convert the tree structured CPTs into conditional probability tables where each row corresponds to each instantiation in the tree and each entry represents a probability distribution in the corresponding instantiation as shown in Tables 3 and 4. We named this conditional probability table as *compact conditional probability table* (compact CPT) since this representation is more compact than the traditional one in terms of the number of rows. We combine the two compact CPTs into a single CPT to be able to make all distinctions that original two CPTs make. Therefore, the compact CPT for factor B is combined with each row of the compact CPT for factor A as shown in Table 5. Since the combined CPT contains redundant rows whose cardinalities conflict each other, we eliminate these inconsistent rows. Finally, the canonical table representation of a new factor C after

multiplying two factors is shown in Table 6. The canonical parameters of the probability distribution can be computed by following the obtained table. For instance, the canonical parameters with $D_1 = 1, D_2 = 2, D_3 = 1$ are computed as follows:

$$\begin{aligned}
K_{a_1}^C &= K_{a_1}^A + K_{a_1}^B \\
\mathbf{h}_{a_1}^C &= \mathbf{h}_{a_1}^A + \mathbf{h}_{a_1}^B \\
g_{a_1}^C &= g_{a_1}^A + g_{a_1}^B.
\end{aligned} \tag{8}$$

Next, let us consider summing out discrete variables D_3 in the compact CPT described in Table 6. We sum up probability distribution to make all distinctions except for D_3 that original table makes. In this example, since D_1 and D_2 of the first and second rows in the table take same values of $D_1 = 1$ and $D_2 = 1$, the corresponding two canonical forms are summed up. Nevertheless, if we need to marginalize factors that contain continuous variables by summing out discrete variables, the resulting distribution is not a Gaussian but a mixture of Gaussians. Therefore, a mixture of Gaussians model should be used to represent the marginalized factors, which can be achieved through Monte Carlo integration (Yuan and Druzdzel, 2006). The problem with this approximation method is that the number of Gaussian components grows with the number of times discrete variables are summed out. In order to avoid such a problem, the mixture of Gaussian is approximated by collapsing it into a single Gaussian.

The marginal distribution over D_1 and D_2 assigns probability distributions to specific events such as $P(D_1 = 1, D_2 = 1)$, $P(D_1 = 2, D_2 = 1)$, $P(D_1 = 1, D_2 = 2)$ and $P(D_1 = 2, D_2 = 2)$. For instance, the marginal probability distribution function $P(D_1 = 1, D_2 = 1)$ can be computed as follows:

$$P(D_1 = 1, D_2 = 1) = C(\mathbf{X}; K_{a_1}^C, \mathbf{h}_{a_1}^C, g_{a_1}^C) + C(\mathbf{X}; K_{a_2}^C, \mathbf{h}_{a_2}^C, g_{a_2}^C). \tag{9}$$

However, as mentioned before, this distribution is not a Gaussian but a mixture of Gaussian and thus it cannot be represented as a canonical form. Instead, we employ the weak

discrete marginalization where the summing operation uses the collapsing operation (Koller and Friedman, 2009). The collapsing operation approximates a mixture of k Gaussian distribution $\mathcal{N}(\mu_i, \Sigma_i)$ by a single Gaussian distribution $\mathcal{N}(\mu_j, \Sigma_j)$. Its mean vector μ_j and covariance matrix Σ_j for new instantiation j are computed as follows:

$$\mu_j = \sum_{i=1}^k p(i|j)\mu_i \quad (10)$$

$$\Sigma_j = \sum_{i=1}^k p(i|j)\Sigma_i + \sum_{i=1}^k p(i|j)(\mu_i - \mu)(\mu_i - \mu)^T \quad (11)$$

where $p(i|j)$ is the conditional probability of the original instantiation i given the new instantiation j and it satisfies $\sum_{i=1}^k p(i|j) = 1$. Therefore, the parameters $(K_{\text{new}}, \mathbf{h}_{\text{new}}, g_{\text{new}})$ of a canonical form $P(D_1 = 1, D_2 = 1)$ which is summed out variable D_3 can be computed as follows:

$$K_{\text{new}} = \Sigma_{\text{new}}^{-1} \quad (12)$$

$$\mathbf{h}_{\text{new}} = \Sigma_{\text{new}}^{-1}\mu_{\text{new}} \quad (13)$$

where

$$\mu_{\text{new}} = p(D_3 = 1|D_1 = 1, D_2 = 1)K_{a_1}^{C-1}\mathbf{h}_{a_1}^C + p(D_3 = 2|D_1 = 1, D_2 = 1)K_{a_2}^{C-1}\mathbf{h}_{a_2}^C \quad (14)$$

$$\begin{aligned} \Sigma_{\text{new}} &= p(D_3 = 1|D_1 = 1, D_2 = 1)K_{a_1}^{C-1} + p(D_3 = 2|D_1 = 1, D_2 = 1)K_{a_2}^{C-1} \\ &\quad + p(D_3 = 1|D_1 = 1, D_2 = 1)(K_{a_1}^{C-1}\mathbf{h}_{a_1}^C - \mu_{\text{new}})(K_{a_1}^{C-1}\mathbf{h}_{a_1}^C - \mu_{\text{new}})^T \\ &\quad + p(D_3 = 2|D_1 = 1, D_2 = 1)(K_{a_2}^{C-1}\mathbf{h}_{a_2}^C - \mu_{\text{new}})(K_{a_2}^{C-1}\mathbf{h}_{a_2}^C - \mu_{\text{new}})^T. \end{aligned} \quad (15)$$

In the case when a set of continuous variables \mathbf{Y} needs to be integrated out, continuous marginalization is carried out after summing out discrete variables. Let us assume that our canonical table consists of a set of canonical forms $C(\mathbf{X}, \mathbf{Y}; K_{a_i}^C, \mathbf{h}_{a_i}^C, g_{a_i}^C)$ indexed by a_i where:

$$K_{a_i} = \begin{bmatrix} K_{\mathbf{X}\mathbf{X}} & K_{\mathbf{X}\mathbf{Y}} \\ K_{\mathbf{Y}\mathbf{X}} & K_{\mathbf{Y}\mathbf{Y}} \end{bmatrix} \quad (16)$$

$$\mathbf{h}_{a_i} = \begin{pmatrix} \mathbf{h}_{\mathbf{X}} \\ \mathbf{h}_{\mathbf{Y}} \end{pmatrix}. \quad (17)$$

The integral over the variables \mathbf{Y} becomes a canonical form $C(\mathbf{X}; K'_{a_i}, \mathbf{h}'_{a_i}, g'_{a_i})$ computed as (Lauritzen, 1992):

$$\begin{aligned} K'_{a_i} &= K_{\mathbf{X}\mathbf{X}} - K_{\mathbf{X}\mathbf{Y}}K_{\mathbf{Y}\mathbf{Y}}^{-1}K_{\mathbf{Y}\mathbf{X}} \\ \mathbf{h}'_{a_i} &= \mathbf{h}_{\mathbf{X}} - K_{\mathbf{X}\mathbf{Y}}K_{\mathbf{Y}\mathbf{Y}}^{-1}\mathbf{h}_{\mathbf{Y}} \\ g'_{a_i} &= g + \frac{1}{2} (\log|2\pi K_{\mathbf{Y}\mathbf{Y}}^{-1}| + \mathbf{h}_{\mathbf{Y}}^T K_{\mathbf{Y}\mathbf{Y}}^{-1} \mathbf{h}_{\mathbf{Y}}) \end{aligned} \tag{18}$$

where $K_{\mathbf{Y}\mathbf{Y}}$ should be positive definite since these equations include the calculation of inverse matrix of $K_{\mathbf{Y}\mathbf{Y}}$.

Above methodology enables us to compute product and marginalization of factors. Compared to the approach that converts tree structure CPTs into the traditional large CPTs, the proposed compact CPTs do not require large computer memory to carry out these operations. Instead, the summing out operations group rows in the CPTs so that all rows in the each group have the same discrete values except for summed out variables and the computational cost of grouping rows increases with $O(n \log n)$. One possible drawback is that the decision tree structures are no longer available once factors are multiplied or marginalized because the compact CPT based operations discard tree structures.

5.3 Belief Propagation Algorithm

In order to implement various kinds of reasoning, we apply the *belief propagation* (BP) algorithm to handle a large hybrid Bayesian network containing large domain of discrete variables. The detailed algorithms of causal, diagnostic and intercausal reasoning are described in Appendices B and C. When entering evidence into Bayesian network, factors need to be modified so that they are consistent with the evidence. The method of inference given the evidence is shown in Appendix D.

It should be noted that the computational task for multiplying incoming messages exponentially increases with the number of incoming messages, when using the sum-product

algorithm. This is the case when the diagnostic, intercausal or mixed inferences are carried out because the LBP requires a large number of calculations for factor multiplication and marginalization. To reduce the computational cost in LBP algorithm, we make use of the property of Bethe cluster graph. In a Bethe cluster graph, the scope of messages covers single variable since any large clusters are connected each other. If we give the evidence to some variable $X_i = e$, the message departed from the corresponding singleton cluster C_i remains the same during iterations. In other word, we can peg the message $\delta_{i \rightarrow k}$ to evidence e and thus we can skip the computation of Eq. (25). The proposed loopy belief propagation algorithm is shown in Table 7.

6 Application Example

6.1 Comparison with SVM and ANN

A simulated example is used to evaluate the validity and performance of the Bayesian networks compared to ANN and SVM. The input and output data are generated from the simple system described in Fig. 9. In this example, $[X_1, X_2, X_3, Y_1, Y_2]$ are input variables while Z is an output variable. Variables are linearly dependent when they are connected with the arcs. Further, all variables are binarized and thus the domain size of each variable is two.

The radial basis kernel function is used for SVM and its parameters are determined automatically using the heuristic method described in the literature (Caputo *et al.*, 2002). As for ANN, the number of hidden variables is determined from cross-validation. The classification accuracies, which are the proportion of true results, along with the number of hidden variables are shown in Table 8. It can be seen that accuracy remains same even when we increase the number of hidden variables. We set the number of hidden variables to be four for ANN.

The computational results are shown in Table 9. When all variables can be observed except for output variables, the accuracies for test data of BN, ANN and SVM are greater than 0.9, which means all methods can accurately predict the output variables. On the other hand, when input data are partially known, both ANN and SVM predict the output variable for test data with lower accuracies of 0.734 and 0.734 respectively. Meanwhile, BN predicts the output variable with the high accuracy of 0.848 even though partial input data can be observed. These computational results demonstrate that BN is superior to SVM and ANN in terms of accuracy with partially observed data.

It is true that SVM is known as one of the strongest classification machine and it works equally to or better than BNs for classification problems with fully observed data. However, as the computational results show, if only partial input data are observed, BN performs better than SVM. In addition, if we need to predict multiple variables, SVM requires multiple models which meet the specific situations, while BN can work for any situations (e.g. fully or partially observed) by using single model. Furthermore, SVM works only for classification problems and thus it is not be able to predict both continuous and discrete variables simultaneously.

6.2 Bayesian Network Model Identification

The real-world plate steel production data are utilized to examine the effectiveness of the proposed Bayesian network based prediction models. All the case studies have been computed on a DELL Optiplex 990 (Intel(R) Core(TM) i-7-2600 CPU, 3.40GHz and 8.0 Gb RAM). First, we use the training data set consisting of 10,000 steel plates to learn the network structure. The Bayesian network structures are identified by maximizing BIC scores as shown in Figs. 10 and 11. The execution time for the network identification was 1,353[s]. Then, we use the training data set consisting of 100,000 steel plates to learn the decision-tree structured CPTs while we use the test data set of another 100,000 plates to evaluate

prediction accuracy. The first and second Bayesian networks are mainly different in the form of the conditional probability of total production time.

6.3 Prediction of Production Loads and Total Production Time

In the first test scenario, we assume that the variables corresponding to the customer demands are known and the other variables are unknown. This situation occurs when persons in the sales department receive orders from customers, since the sales planning staff need to know the production loads and the process time of these orders as required to satisfy the customer demands and within constraints of the production capacity. The task of this inference is to predict the effects of production loads and a process time given the causes of the customer demands, which means that the type of inference is causal reasoning. In order to compute the true probability distributions which are required for comparison to the prediction, we classify the test data into production groups which contain plates that have the same customer specifications and operating conditions and then compute the probability distribution for each production group. The production groups containing more than 100 plates are chosen for comparison while the other production groups are not selected because we cannot compute the reliable probability distributions from a small number of samples. The total number of the production groups that are used for comparison is 106.

First, we predict the production loads and the production time by using the first Bayesian network structure shown in Fig. 10. Fig. 12 shows the scatter plots between the predicted and true probabilities with production loads being 2. Further, the Kullback-Leibler (KL) divergence, which measures the difference between the predicted probability distributions and actual ones is shown in Fig. 13. It can be observed that the difference between the predicted and true probability distributions of the variables of process A, process B, process C, process D, process E, process F and process G are relatively small while the predicted probabilities with process H and process I values being 2 are zero in all production

groups. Moreover, the scatter plots between the predicted and true parameters of means and standard deviations of the production time are shown in 14 , and their KL divergences are provided in Fig. 15. It can be seen that the means of the predicted distributions are relatively close to the true means but the variances of the predicted distributions largely differ from the actual ones. Somewhat inaccurate prediction of the production loads of process E, process F, process H and process I can be considered a cause of these differences in the production time.

Next, the probability distributions of the production loads and the production time are computed via the second type of Bayesian network, shown in Fig. 11. The difference between the two structures is that the nodes associated with the production time at each production process are added between the nodes of production loads and total production time in the second type on Bayesian network. Similarly to the first model, we provide the scatter plots of the predicted and true parameters of means and standard deviations and their KL divergences in Figs. 16 and 17, respectively. It can be observed that the prediction accuracy does not improve at all from the first Bayesian network. The cause of inaccurate prediction of the variances of the total production time is the low accurate prediction of the production loads.

In the second test case, we consider the situation where the variables associated with both customer demands and operating conditions are known and the variables of both production loads and process time are unknown. In this situation, we want to predict the probability distributions of the production loads and the process time given all information except for these target variables. Such situation often arises at the production scheduling stage. Similarly to the previous test scenario, the types of inference is causal reasoning and we make use of the same decision-tree structured CPTs as the previous test case. Fig. 18 shows the scatter plots between the predicted and actual probability of the production loads. In addition, their KL divergences are provided in Fig. 19. It can be observed that the

predicted and true probability distributions are almost same. In particular, the accuracy of the predicted distributions of loads of process D, process E, process F, process G, process H and process I loads are significantly improved compared to the first test scenario. All of their RMSEs are less than 0.1 and the correlations are also much higher than those in the first test scenario. This is mainly because we know the additional values of the operating conditions that are considered having a significant influence on these production loads. However, in spite of the accurate prediction of production loads, the means of the predicted probability distributions of the production times are not improved as shown in Figs. 20 and 21.

The second type of Bayesian network shown as Fig. 11 is also used to predict the production time. The scatter plots between the predicted and actual parameters of means and standard variances and their KL divergences are given in Figs. 16 and 17. One can readily see that the KL divergences between the predicted and actual probability distribution are smaller than all the above results. In particular, the predicted means are very close to the actual ones. As for the accuracy of the predicted standard variances, the second Bayesian network model can accurately predict the standard deviations in most production groups. In some production groups, the differences between the predicted and actual standard deviations are somewhat large. In order to improve further, it may be desirable to add other nodes of variables such as the amount of in-process inventory, the number of workers and the failure rate of each machine in Bayesian networks, which should have an impact on the process time.

All the results indicate that even though our Bayesian network includes large domain of discrete variables, the causal reasoning can be carried out by using the proposed decision-tree structured conditional probability table representations. Moreover, a single Bayesian network model can carry out the most plausible reasoning and its accuracy depends on using the values of known variables. This means that we can avoid having multiple specific

models tailored for specific purposes. Therefore, the proposed Bayesian network model satisfies the properties required for plate steel production scheduling and planning.

6.4 Prediction of Operating Conditions

Up to now we have dealt with prediction of the probability distributions of the production loads and the total production time. In this last test case, we assume that the values of the customer demands and production loads are known while the operating conditions are unknown. Such situation arises when we want to analyze the relationship between the production loads and the operating conditions, or we need to design the operating conditions so that both the customer demands and production capacity constraints can be satisfied. Unlike the previous test scenarios, the type of the inference is mixed reasoning and thus the loopy belief propagation algorithm is carried out. Similarly to the previous two scenarios, we evaluate the prediction accuracy for each production of steel plates with the same customer demands and production loads. We select the production groups containing more than 100 plates to compute the true probability distributions, which are then used for comparison with the predicted probability distributions. The number of production groups that are used in this comparison is 90. The probability of cardinality of the predicted and true operating conditions is shown in Fig. 24. In addition, their KL divergences are shown in Fig. 25. One can readily see that most KL divergences are small; hence, the proposed inference algorithm can accurately predict the probability distributions of unknown variables in the mixed reasoning. Above results demonstrate that the proposed Bayesian inference method can carry out the mixed reasoning even though Bayesian networks have large domain of discrete variables.

Average execution times for all the above test scenarios are provided in Table 10. Comparing the execution time of the first and the second Bayesian networks, we note that the average execution time of the second model is less than 1[s], which is much smaller than

that of the first Bayesian network model. That is because in the first model, the node of total production time is connected to the all nodes of the production loads as their child and thus we need a large number of multiplication of factors to compute the final potential of the production time, which leads to high computational effort since the computational cost of factor multiplication increases with the number of variables contained in the factor. For the mixed reasoning, the average execution time is less than 2[s], which is small enough to use it in practical applications.

7 Conclusion

In this study, we propose the Bayesian network models for prediction of the production loads and the production time in manufacturing of steel plates. In order to identify the network structure from historical process data, we search for the graph such that the maximized Bayesian scores are obtained. Since the network contains both continuous and discrete system variables and some of discrete variables have large cardinalities, the decision-tree structured CPTs based hybrid Bayesian inference technique is proposed. Once the Bayesian network structure is constructed, we compute the context-specific conditional probability tables represented by decision trees from historical process data set by using classification tree algorithm. Then, operators for computing multiplication and marginalization of factors represented by decision-tree structure based CPTs are developed. The Bayesian inference is carried out using belief propagation algorithms with the proposed multiplying and marginalizing operators. The causal reasoning is carried out by means of the sum-product algorithm whose direction is downstream from the top node variables to the bottom nodes variables. Other types of reasoning are carried out by the loopy belief propagation.

Real life steel production process data have been used to examine the effectiveness of the proposed inference algorithms. Even though our Bayesian network contains large domain of discrete variable nodes, the results show that the proposed algorithm can be successfully

applied to industrial scale Bayesian networks and it can predict the probability distributions of unobserved variables such as the production loads and the production times in the steel plate production.

Results of this work will be applied to production planning and scheduling in manufacturing of steel plates.

Appendix A Structure Learning

Several kinds of scoring functions have been developed. Most of them have a decomposable score function as follows:

$$\text{score}(\mathcal{G}) = \sum_{i=1}^I \text{score}(x_i, \text{pa}_{\mathcal{G}}(x_i)) \quad (19)$$

where $\text{pa}_{\mathcal{G}}(x_i)$ are the parent nodes of x_i given graph \mathcal{G} . Among various kinds of score functions, the BIC score (for Bayesian information criterion) is widely used; it is defined as

$$\text{score}_{\text{BIC}}(\mathcal{G} : \mathcal{D}) = \ell(\hat{\theta}_{\mathcal{G}} : \mathcal{D}) - \frac{\text{Log}M}{2} \text{Dim}[\mathcal{G}] \quad (20)$$

where $\hat{\theta}_{\mathcal{G}}$ denotes the maximum likelihood parameters given a graph \mathcal{G} , $\ell(\hat{\theta}_{\mathcal{G}} : \mathcal{D})$ denotes the logarithm of the likelihood function, M is the number of samples, and $\text{Dim}[\mathcal{G}]$ is the model dimension that is equal to the number of parent nodes. The term of model dimension is introduced in the score function in order to handle a tradeoff between the likelihood and model complexity. Since the BIC scores are decomposable, they can be restated as follows:

$$\text{score}_{\text{BIC}}(\mathcal{G} : \mathcal{D}) = M \sum_{i=1}^N \mathbf{I}(x_i; \text{pa}_{\mathcal{G}}(x_i)) - M \sum_{i=1}^N H(x_i) - \frac{\text{Log}M}{2} \text{Dim}[\mathcal{G}] \quad (21)$$

where $\mathbf{I}(x_i; \text{pa}_{\mathcal{G}}(x_i))$ is the mutual information between x_i and $\text{pa}_{\mathcal{G}}(x_i)$, and $H(x_i)$ is the marginal entropy of x_i .

Score decomposability is important property for network structure learning because if the decomposability is satisfied, a local change such as adding, deleting or reversing an

edge in the structure does not change the score of other parts of the structure that remain same. It also should be pointed out that the mutual information term grows linearly in M while the model complexity term grows logarithmically, and thus the larger M leads to an obtained graph which better represents data (Koller and Friedman, 2009).

With the score function defined as above, the optimal graph \mathcal{G}^* can be computed as follows:

$$\mathcal{G}^* = \operatorname{argmax}_{\mathcal{G}} \operatorname{score}_{\text{BIC}}(\mathcal{G} : \mathcal{D}). \quad (22)$$

Appendix B Causal Reasoning

The computational task of causal reasoning is to estimate the downstream effects of factors given the evidence of their ancestor variables. In our example, causal reasoning predicts the probability distributions of production loads and total production time from the evidence of customer demands, operating conditions or both. Conditional probability can be computed through a local message passing algorithm known as sum-product algorithm in downstream direction from the root nodes to the leaf nodes. Since our Bayesian network contains both discrete and continuous variables, a final potential $\tilde{P}_{\Phi}(\mathbf{C}_i)$ of factor $\mathbf{C}_i = \mathbf{A}_i \cup \mathbf{X}_i$ is computed using sum and integral operators as follows:

$$\tilde{P}_{\Phi}(\mathbf{C}_i) = \int \sum_{\mathbf{A}_i - \mathbf{A}_i} \psi_i \prod_{k \in \text{pa}(i)} P_{\Phi}(\mathbf{C}_k) dx_1 dx_2 \dots dx_M \quad (23)$$

where:

- ψ_i is the initial potential of \mathbf{C}_i , $P_{\Phi}(\mathbf{C}_k)$ is the normalized final potential of \mathbf{C}_k ,
- $x_m \in \{\mathbf{X}_i - X_i\}$ are variables that we integrate out, and
- X_i or A_i is the child node of the factor \mathbf{C}_i where if the child node variable is continuous, $A_i = \emptyset$, otherwise $X_i = \emptyset$.

First, we compute the initial potentials ψ_i for all factors. Each node in the Bayesian network is assigned one factor and thus the number of factors is equal to the number of nodes. A factor belonging to a set of top root nodes is assigned a singleton factor that contains a variable of the corresponding node. Since evidence is given to all singleton factors in the causal reasoning, their initial and final potentials have cardinality of one and their probabilities are always 1. Note that a factor corresponding to a node that is not placed on the top root node is assigned a non-singleton factor that contains a variable of the corresponding node and its all parent nodes. Therefore, the initial potentials for non-singleton factors are equal to the conditional probability distributions of its corresponding variable given its parent node variables. After computing initial potentials of all factors, the sum-product algorithm proceeds in the downstream direction from the top factors to the bottom factors in order to compute the final potentials of all factors. The causal reasoning algorithm searches for the non-singleton factor for which all parent nodes are assigned final potentials and then computes the final potential for that non-singleton factor by using Eq. (23). Since the final potential $\tilde{P}_\Phi(\mathbf{C}_i)$ in Eq. (23) is not normalized, we need to normalize it and obtain $P_\Phi(\mathbf{C}_i)$ every time after computing the final potentials. This algorithm continues computing final potentials using the sum-product algorithm until final potentials of all factors are computed. Since the computation of final potentials is carried out from top node factors to bottom node factors, the probability propagation directions are the same as the arcs in Bayesian networks. The computed final potentials $P_\Phi(\mathbf{C}_k)$ are equal to the conditional probability distributions of the corresponding variable given the evidence.

Appendix C Diagnostic and Intercausal Reasoning

While causal reasoning is carried out by simply applying the sum-product algorithm from the top nodes to the bottom nodes, conditional probability for diagnostic, intercausal and

mixed reasoning can be computed by iteratively carrying out the sum-product algorithm until convergence in the loopy belief propagation (LBP). LBP schemes use a cluster graph rather than a clique tree that is used for VE (Murphy *et al.*, 1999). Since the constraints defining the clique tree are indispensable for exact inference, the answers of message passing scheme are approximate queries in LBP.

First, we create the singleton factors for all nodes and then each singleton factor is assigned an initial potential of the prior probability of the corresponding variable. Also, we create the intermediate factors for all nodes that have parent nodes and then each factor is assigned an initial potential of the conditional probability distribution of the corresponding variable given its parent variables. Therefore, the number of factors is larger than the number of nodes. Given the cluster graph, we assign each factor ϕ_k to a cluster $\mathbf{C}_{\alpha(k)}$ so that $\text{scope}(\phi_k) \subseteq \mathbf{C}_{\alpha(k)}$. Then the initial potentials ψ_i can be computed as follows:

$$\psi_i = \prod_{k:\alpha(k)=i} \phi_k. \quad (24)$$

In LBP, one factor communicates with another factor by passing messages between them. The messages have canonical form representation and their potentials are initialized as $\{K, \mathbf{h}, g\} = \{1, 0, 0\}$. A message from cluster $\mathbf{C}_i = \mathbf{A}_i \cup \mathbf{X}_i$ to another factor $\mathbf{C}_j = \mathbf{A}_j \cup \mathbf{X}_j$ is computed using sum and integral operators as follows:

$$\delta_{i \rightarrow j} = \int \sum_{\mathbf{A}_i - \mathbf{S}_{i,j}} \psi_i \prod_{k \in \text{Nb}_i - \{j\}} \delta_{k \rightarrow i} dx_1 dx_2 \dots dx_M \quad (25)$$

where:

- $x_m \in \{\mathbf{X}_i - \mathbf{S}'_{i,j}\}$ are integrated variables,
- $\mathbf{S}_{i,j} = \mathbf{A}_i \cap \mathbf{A}_j$ is the sepset of discrete variables,
- $\mathbf{S}'_{i,j} = \mathbf{X}_i \cap \mathbf{X}_j$ is the sepset of continuous variables, and
- Nb_i is the set of indices of factors that are neighbors of \mathbf{C}_i .

The messages are updated by using Eq. (25) in accordance with some message passing schedule until the canonical parameters of all messages are converged. Using all incoming messages, a final potential $\tilde{P}_\Phi(\mathbf{C}_i)$ can be computed by multiplying them with the initial potential as follows:

$$\tilde{P}_\Phi(\mathbf{C}_i) = \psi_i \prod_{k \in \text{Nb}_i} \delta_{k \rightarrow i}. \quad (26)$$

Similar to the causal reasoning, the final potential $\tilde{P}_\Phi(\mathbf{C}_i)$ needs normalization.

In order to carry out the LBP algorithm, we have to create a cluster graph in advance such that it satisfies the family preservation property. For this purpose, the *Bethe cluster* graph which is a bipartite graph and holds the family preservation property is widely used. The first layer of the Bethe cluster graph consists of large clusters $\mathbf{C}_k = \text{scope}(\phi_k)$ while the second layer consists of singleton clusters X_i . Then, edges are placed between a large cluster \mathbf{C}_k and a singleton cluster X_i if $X_i \in \mathbf{C}_k$. Fig. 8 shows an example of the Bethe cluster graph. In this example, the Bethe cluster graph has 6 singleton clusters $\{A\}, \{B\}, \{C\}, \{D\}, \{E\}, \{F\}$ and 3 large clusters $\{A, B, D\}, \{B, C, E\}, \{D, E\}$. For instance, due to $\{B\} \in \{A, B, D\}$ and $\{B\} \in \{B, C, E\}$, the edges between cluster 2 and 7 and between cluster 2 and 8 are placed.

Appendix D Belief Propagation with Evidence

When initializing the factor ψ_i given the discrete evidence $e \in E \cap \Delta$, we delete all sets of canonical parameters K, \mathbf{h}, g in the factor ψ_i that conflict the evidence e .

Next, let us consider initializing the factor ψ_i given the continuous evidence $e \in E \cap \Gamma$. If the factor does not have any discrete variables, a canonical form of the factor ψ_i is reduced to a context representing the evidence e . In Eq. (16), we set $\mathbf{Y} = y$ with y being the value of the evidence e , then the new canonical form given continuous evidence is described as

follows (Lauritzen, 1992):

$$\begin{aligned}
K' &= K_{\mathbf{X}\mathbf{X}} \\
\mathbf{h}' &= \mathbf{h}_{\mathbf{X}} - K_{\mathbf{X}\mathbf{Y}}y \\
g' &= g + \mathbf{h}_{\mathbf{Y}}^T y - \frac{1}{2}y^T \mathbf{K}_{\mathbf{Y}\mathbf{Y}}y.
\end{aligned} \tag{27}$$

If the factor has discrete variables and does not have a continuous variable except for $\text{scope}(E)$, the canonical parameter g_{a_i} is modified for each instantiation a_i as follows:

$$g_{a_i} = -\frac{1}{2}y K_{a_i} y + \mathbf{h}_{a_i}^T y + g_{a_i}. \tag{28}$$

The canonical parameters K_{a_i} and h_{a_i} become vacuous because the new factor contains no continuous variable. If the factor contains both discrete and continuous variables except for $\text{scope}(E)$, the parameters of the canonical form are computed for each instantiation a_i using Eq. (27). After modifying all factors so that all of them are consistent with the evidence, the reasoning algorithms mentioned above can be carried out.

References

- Ashayeria, J., R.J.M. Heutsa, H.G.L. Lansdaalb and L.W.G. Strijbosch (2006). Cyclic productioninventory planning and control in the pre-deco industry: A case study. *Int. J. Prod. Econ.* **103**, 715–725.
- Bishop, C.M. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag, New York, NY.
- Boutilier, C., N. Friedman, M. Goldszmidt and D. Koller (1996). Context-specific independence in Bayesian networks. In: *Proceedings of UAI-96*. pp. 115–123.
- Breiman, L., J.H. Friedman, R.A. Olshen and C.G. Stone (1984). *Classification and Regression Trees*. Wadsworth International Group. Belmont.
- Caputo, B., K. Sim, F. Furesjo and A. Smola (2002). Appearance-based object recognition using svms: which kernel should i use?. In: *Proceedings of NIPS workshop on Statistical methods for computational experiments in visual processing and computer vision*.
- Chickering, D.M. (1996). Learning bayesian networks is np-complete. In: *Learning from Data: Artificial Intelligence and Statistics*. Chap. 12. Springer-Verlag.
- Chickering, D.M., D. Heckerman and C. Meek (1997). A bayesian approach to learning Bayesian networks with local structure. In: *Proceedings of UAI-97*. pp. 80–89.
- Cobb, B.R., R. Rumi and A. Salmeron (2007). *Bayesian Network Models with Discrete and Continuous Variables*. SpringerVerlag. Berlin.
- DesJardins, M. and P. Rathod (2008). Learning structured Bayesian networks: Combining abstraction hierarchies and tree-structured conditional probability tables. *Comput. Intell.* **24**, 1–22.

- Friedman, N. and M. Goldszmidt (1996). Learning Bayesian networks with local structure. In: *Proceedings of UAI-96*. pp. 252–262.
- Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Comput. Oper. Res.* **13**, 533–549.
- I. Harjunkoski, I.E. Grossmann (2001). A decomposition approach for the scheduling of a steel plant production. *Comput. Chem. Eng.* **25**, 1647–1660.
- Koller, D. and N. Friedman (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.
- Koller, D., U. Lerner and D. Angelov (1999). A general algorithm for approximate inference and its application to hybrid bayes nets. In: *Proceedings of UAI-99*. pp. 324–333.
- Kozlov, A. and D. Koller (1997). Nonuniform dynamic discretization in hybrid networks. In: *Proceedings of UAI-97*. pp. 314–325.
- Lauritzen, S. (1992). Propagation of probabilities, means and variances in mixed graphical association models. *J. Amer. Stat. Assoc.* **87**, 1098–1108.
- Lauritzen, S.L. and F. Jensen (2001). Stable local computation with conditional gaussian distribution. *Stat. Comput.* **11**, 191–203.
- Lauritzen, S.L. and N. Wermuth (1989). Graphical models for associations between variables, some of which are qualitative and some quantitative. *Ann. Stat.* **17**, 31–57.
- Moon, S. and A.N. Hrymak (1999). Scheduling of the batch annealing processdeterministic case. *Comput. Chem. Eng.* **23**, 1193–1208.
- Moral, S., R. Rumi and A. Salmern (2001). *Mixtures of Truncated Exponentials in Hybrid Bayesian Networks*. SpringerVerlag. Berlin.

- Murphy, K.P., Y. Weiss and M.I. Jordan (1999). Nonuniform dynamic discretization in hybrid networks. In: *Proceedings of UAI-99*. pp. 467–475.
- Nishioka, K., Y. Mizutani, H. Ueno, H. Kawasaki and Y. Baba (2012). Toward the integrated optimization of steel plate production process : A proposal for production control by multi-scale hierarchical modeling (in Japanese). *Synthesiology* **5**, 98–112.
- Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan-Kaufmann. San Francisco.
- Poole, D. and N.L. Zhang (2003). Exploiting contextual independence in probabilistic inference. *J. Artif. Intell. Res.* **18**, 263–313.
- Rumi, R. and A. Salmeron (2007). Approximate probability propagation with mixtures of truncated exponentials. *Int. J. Approx. Reason.* **45**, 191–210.
- Santos, C.A., J.A. Spim and A. Garci (2003). Mathematical modeling and optimization strategies (genetic algorithm and knowledge base) applied to the continuous casting of steel. *Eng. Appl. Artif. Intel.* **16**, 511–527.
- Sharma, R. and D. Poole (2003). Efficient inference in large discrete domains. In: *Proceedings of UAI-2003*. pp. 535–542.
- Tang, L., J. Liu, A. Rong and Z. Yang (2000). A mathematical programming model for scheduling steelmaking continuous casting production. *Eur. J. Oper. Re.* **120**, 423–435.
- Yuan, C. and M. J. Druzdzel (2006). Hybrid loopy belief propagation. In: *Proceedings of the Third European Workshop on Probabilistic Graphical Models (PGM'06)*. pp. 317–324.

Table 1: System variables of steel production process

| Variable No. | Variable description | Domain size |
|--------------|-----------------------|-----------------|
| 1 | Customer demand A | 12 |
| 2 | Customer demand B | 2 |
| 3 | Customer demand C | 20 |
| 4 | Customer demand D | 20 |
| 5 | Customer demand E | 20 |
| 6 | Customer demand F | 10(discretized) |
| 7 | Customer demand G | 10(discretized) |
| 8 | Customer demand H | 10(discretized) |
| 9 | Customer demand I | 10(discretized) |
| 10 | Operating condition A | 5 |
| 11 | Operating condition B | 2 |
| 12 | Operating condition C | 2 |
| 13 | Operating condition D | 2 |
| 14 | Operating condition E | 2 |
| 15 | Operating condition F | 2 |
| 16 | Operating condition G | 2 |
| 17 | Production load A | 2 |
| 18 | Production load B | 2 |
| 19 | Production load C | 2 |
| 20 | Production load D | 2 |
| 21 | Production load E | 2 |
| 22 | Production load F | 2 |
| 23 | Production load G | 2 |
| 24 | Production load H | 2 |
| 25 | Production load I | 2 |
| 26 | Process time | continuous |

Table 2: Algorithm 1. Network identification algorithm

```

Procedure Network-Identification (
   $\mathcal{D}$  // training data set
   $\mathcal{G}_0$  // initial network structure
   $\mathcal{R}$  // a set of fixed root nodes
   $\mathcal{L}$  // a set of fixed leaf nodes
   $\prec$  // a set of incomplete orderings
  global TabuList // tabu list
)
1 TabuList  $\leftarrow \emptyset$ 
2  $\mathcal{G} \leftarrow \text{Search}(\mathcal{G}_0, \mathcal{R}, \mathcal{L}, \prec)$ 
3  $\mathcal{G}_{\text{best}} \leftarrow \mathcal{G}$ 
4 for  $i = 1, \dots$  until convergence
5    $\mathcal{G} \leftarrow \text{Search}(\mathcal{G}, \mathcal{R}, \mathcal{L}, \prec)$ 
6   if  $\text{score}_{\text{BIC}}(\mathcal{G} : \mathcal{D}) > \text{score}_{\text{BIC}}(\mathcal{G}_{\text{best}} : \mathcal{D})$ 
7      $\mathcal{G}_{\text{best}} \leftarrow \mathcal{G}$ 
8 return  $\mathcal{G}_{\text{best}}$ 

Procedure Search (
   $\mathcal{G}$  // network structure
   $\mathcal{R}$  // a set of fixed root nodes
   $\mathcal{L}$  // a set of fixed leaf nodes
   $\prec$  // a set of incomplete orderings
)
1 while new graph is not obtained
2   Select indices  $i$  and  $j$  ( $i \neq j$ ) at random
3   if edge  $i \rightarrow j$  exists in  $\mathcal{G}$ 
4     Delete edge  $i \rightarrow j$  or reverse edge  $i \rightarrow j$  at random and then get  $\mathcal{G}_{\text{new}}$ 
5   else if edge  $j \rightarrow i$  exists in  $\mathcal{G}$ 
6     Delete edge  $j \rightarrow i$ , add new edge  $i \rightarrow j$  and then get  $\mathcal{G}_{\text{new}}$ 
7   else
8     Add new edge  $i \rightarrow j$  and then get  $\mathcal{G}_{\text{new}}$ 
9   if  $\mathcal{G}_{\text{new}}$  is cyclic
10    continue
11  if  $\mathcal{G}_{\text{new}}$  is in TabuList
12    continue
13  if  $\mathcal{G}_{\text{new}}$  conflicts either  $\mathcal{R}$  or  $\mathcal{L}$  or  $\prec$ 
14    continue
15  Add  $\mathcal{G}_{\text{new}}$  to TabuList
16 return  $\mathcal{G}_{\text{new}}$ 

```

Table 3: Compact CPT for Factor A

| D_1 | D_2 | Probability Distribution |
|-------|-------|---|
| 1 | 1 | $C(\mathbf{X}; K_{a_1}^A, \mathbf{h}_{a_1}^A, g_{a_1}^A)$ |
| 1 | 2 | $C(\mathbf{X}; K_{a_2}^A, \mathbf{h}_{a_3}^A, g_{a_2}^A)$ |
| 2 | 1,2 | $C(\mathbf{X}; K_{a_3}^A, \mathbf{h}_{a_2}^A, g_{a_3}^A)$ |

Table 4: Compact CPT for Factor B

| D_2 | D_3 | Probability Distribution |
|-------|-------|---|
| 1 | 1 | $C(\mathbf{X}; K_{a_1}^B, \mathbf{h}_{a_1}^B, g_{a_1}^B)$ |
| 1 | 2 | $C(\mathbf{X}; K_{a_2}^B, \mathbf{h}_{a_2}^B, g_{a_2}^B)$ |
| 2 | 1,2 | $C(\mathbf{X}; K_{a_3}^B, \mathbf{h}_{a_3}^B, g_{a_3}^B)$ |

Table 5: Combined Compact CPT of Factor A and B

| Factor A | | Factor B | | Probability Distribution |
|----------|-------|----------|-------|---|
| D_1 | D_2 | D_2 | D_3 | |
| 1 | 1 | 1 | 1 | $C(\mathbf{X}; K_{a_1}^C, \mathbf{h}_{a_1}^C, g_{a_1}^C) = C(\mathbf{X}; K_{a_1}^A, \mathbf{h}_{a_1}^A, g_{a_1}^A) \cdot C(\mathbf{X}; K_{a_1}^B, \mathbf{h}_{a_1}^B, g_{a_1}^B)$ |
| | | 1 | 2 | $C(\mathbf{X}; K_{a_2}^C, \mathbf{h}_{a_2}^C, g_{a_2}^C) = C(\mathbf{X}; K_{a_1}^A, \mathbf{h}_{a_1}^A, g_{a_1}^A) \cdot C(\mathbf{X}; K_{a_2}^B, \mathbf{h}_{a_2}^B, g_{a_2}^B)$ |
| | | 2 | 1,2 | inconsistent |
| 1 | 2 | 1 | 1 | inconsistent |
| | | 1 | 2 | inconsistent |
| | | 2 | 1,2 | $C(\mathbf{X}; K_{a_3}^C, \mathbf{h}_{a_3}^C, g_{a_3}^C) = C(\mathbf{X}; K_{a_2}^A, \mathbf{h}_{a_2}^A, g_{a_2}^A) \cdot C(\mathbf{X}; K_{a_3}^B, \mathbf{h}_{a_3}^B, g_{a_3}^B)$ |
| 2 | 1,2 | 1 | 1 | $C(\mathbf{X}; K_{a_4}^C, \mathbf{h}_{a_4}^C, g_{a_4}^C) = C(\mathbf{X}; K_{a_3}^A, \mathbf{h}_{a_3}^A, g_{a_3}^A) \cdot C(\mathbf{X}; K_{a_1}^B, \mathbf{h}_{a_1}^B, g_{a_1}^B)$ |
| | | 1 | 2 | $C(\mathbf{X}; K_{a_5}^C, \mathbf{h}_{a_5}^C, g_{a_5}^C) = C(\mathbf{X}; K_{a_3}^A, \mathbf{h}_{a_3}^A, g_{a_3}^A) \cdot C(\mathbf{X}; K_{a_2}^B, \mathbf{h}_{a_2}^B, g_{a_2}^B)$ |
| | | 2 | 1,2 | $C(\mathbf{X}; K_{a_6}^C, \mathbf{h}_{a_6}^C, g_{a_6}^C) = C(\mathbf{X}; K_{a_3}^A, \mathbf{h}_{a_3}^A, g_{a_3}^A) \cdot C(\mathbf{X}; K_{a_2}^B, \mathbf{h}_{a_2}^B, g_{a_2}^B)$ |

Table 6: Compact CPT for Factor C

| D_1 | D_2 | D_3 | Probability Distribution |
|-------|-------|-------|---|
| 1 | 1 | 1 | $C(\mathbf{X}; K_{a_1}^C, \mathbf{h}_{a_1}^C, g_{a_1}^C) = C(\mathbf{X}; K_{a_1}^A, \mathbf{h}_{a_1}^A, g_{a_1}^A) \cdot C(\mathbf{X}; K_{a_1}^B, \mathbf{h}_{a_1}^B, g_{a_1}^B)$ |
| 1 | 1 | 2 | $C(\mathbf{X}; K_{a_2}^C, \mathbf{h}_{a_2}^C, g_{a_2}^C) = C(\mathbf{X}; K_{a_1}^A, \mathbf{h}_{a_1}^A, g_{a_1}^A) \cdot C(\mathbf{X}; K_{a_2}^B, \mathbf{h}_{a_2}^B, g_{a_2}^B)$ |
| 1 | 2 | 1,2 | $C(\mathbf{X}; K_{a_3}^C, \mathbf{h}_{a_3}^C, g_{a_3}^C) = C(\mathbf{X}; K_{a_2}^A, \mathbf{h}_{a_2}^A, g_{a_2}^A) \cdot C(\mathbf{X}; K_{a_3}^B, \mathbf{h}_{a_3}^B, g_{a_3}^B)$ |
| 2 | 1 | 1 | $C(\mathbf{X}; K_{a_4}^C, \mathbf{h}_{a_4}^C, g_{a_4}^C) = C(\mathbf{X}; K_{a_3}^A, \mathbf{h}_{a_3}^A, g_{a_3}^A) \cdot C(\mathbf{X}; K_{a_1}^B, \mathbf{h}_{a_1}^B, g_{a_1}^B)$ |
| 2 | 1 | 2 | $C(\mathbf{X}; K_{a_5}^C, \mathbf{h}_{a_5}^C, g_{a_5}^C) = C(\mathbf{X}; K_{a_3}^A, \mathbf{h}_{a_3}^A, g_{a_3}^A) \cdot C(\mathbf{X}; K_{a_2}^B, \mathbf{h}_{a_2}^B, g_{a_2}^B)$ |
| 2 | 2 | 1,2 | $C(\mathbf{X}; K_{a_6}^C, \mathbf{h}_{a_6}^C, g_{a_6}^C) = C(\mathbf{X}; K_{a_3}^A, \mathbf{h}_{a_3}^A, g_{a_3}^A) \cdot C(\mathbf{X}; K_{a_2}^B, \mathbf{h}_{a_2}^B, g_{a_2}^B)$ |

Table 7: Algorithm 2. Hybrid loopy belief propagation algorithm

```

Procedure Hybrid-LBP (
   $\Phi$  // Set of factors
   $\mathbf{e}$  // Evidence
   $\mathcal{G}_{\text{cluster}}$  // Bethe cluster graph
)
1 Set  $\mathcal{E}$  to be set of edges in  $\mathcal{G}_{\text{cluster}}$ 
2 Initialize all factors so that all of them are consistent with the evidence  $\mathbf{e}$ 
3 Initialize all messages as  $\{K_{a_i}, \mathbf{h}_{a_i}, g_{a_i}\} = \{1, 0, 0\}$  for each instantiation  $a_i$ 
4 while any one of messages are not converged
5   Select  $(i, j) \in \mathcal{E}$  by following message schedule
6   if  $\text{scope}(\psi_i) = \text{scope}(e)$ 
7     continue
8   Update message  $\mu_{i \rightarrow j}$  from Eq. (25)
9   Compute final potential  $\tilde{P}_{\Phi}(\mathbf{C}_i)$  from (26)
10  Normalize  $\tilde{P}_{\Phi}(\mathbf{C}_i)$  and get  $P_{\Phi}(\mathbf{C}_i)$ 
11 return  $P_{\Phi}(\mathbf{C}_i)$ 

```

Table 8: Results of the cross-validation (ANN)

| The number of hidden variables | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|--------------------------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| Accuracy | 0.928 | 0.928 | 0.928 | 0.930 | 0.926 | 0.930 | 0.928 | 0.930 | 0.928 | 0.928 |

Table 9: Results of BN, ANN and SVM for the simple system

| known variables | BN | ANN | SVM |
|--|-------|-------|-------|
| $[X_1, X_2, X_3, Y_1, Y_2]$ (Fully observed) | 0.970 | 0.958 | 0.958 |
| $[X_1, X_2, X_3]$ Partially observed) | 0.838 | 0.734 | 0.734 |

Table 10: Average of execution time for inference [s]

| Case No. | Reasoning Type | First model | Second model |
|----------|----------------|-------------|--------------|
| 1 | causal | 4.33 | 0.85 |
| 2 | causal | 4.06 | 0.56 |
| 3 | mixed | 1.52 | |

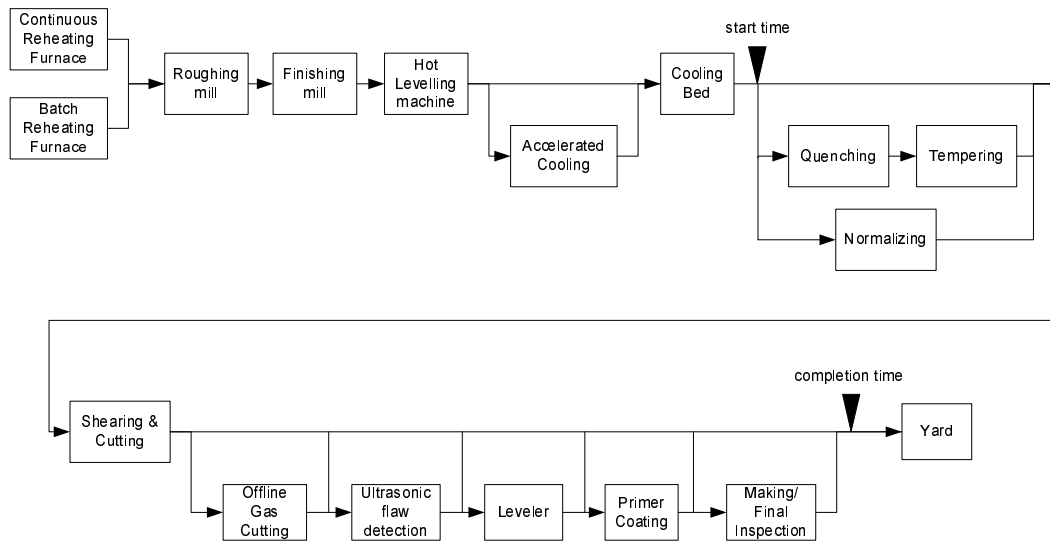


Figure 1: The example process

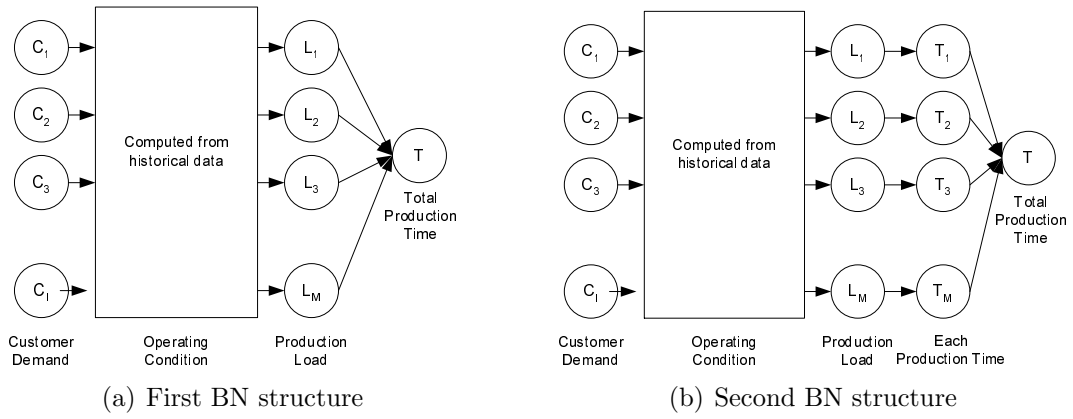


Figure 2: Two types of Bayesian network structure

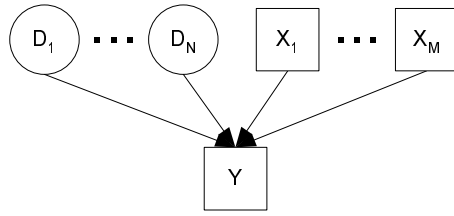


Figure 3: Bayesian network that contains continuous child node

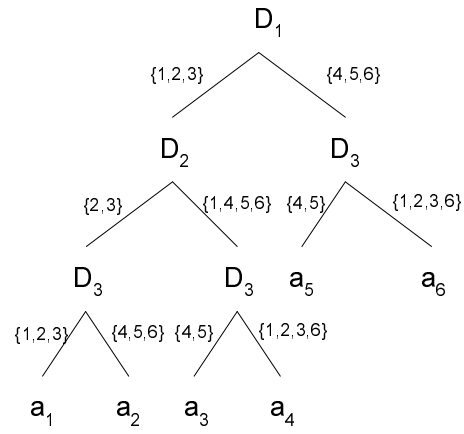


Figure 4: Decision tree of a factor that contains continuous child node

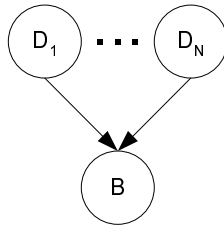


Figure 5: Bayesian network that contains discrete child node

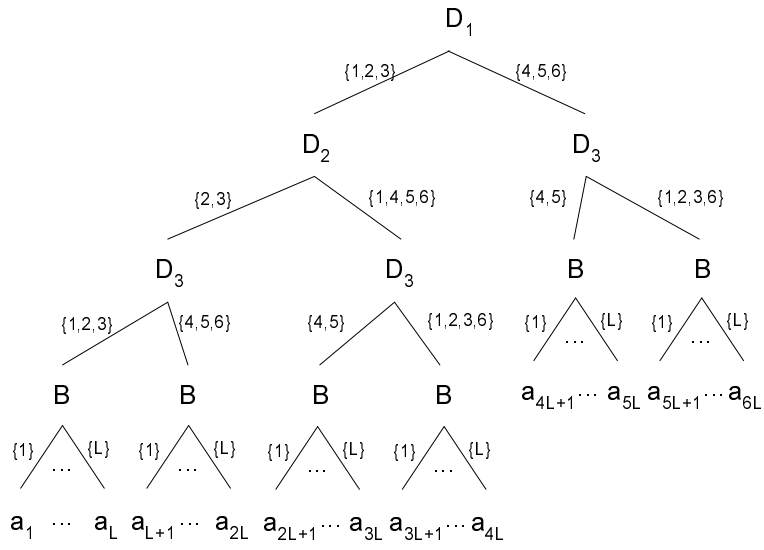


Figure 6: Decision tree of a factor that contains discrete child node

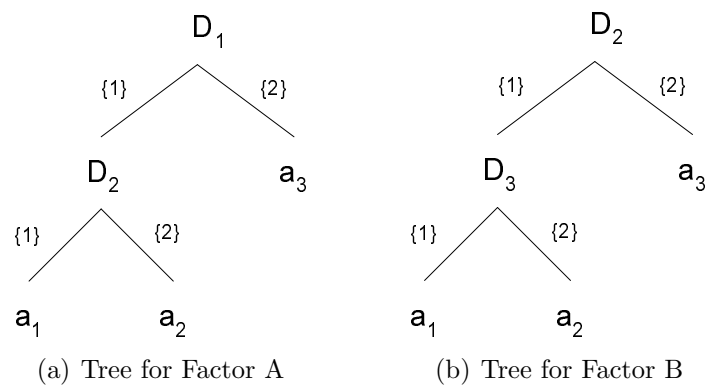
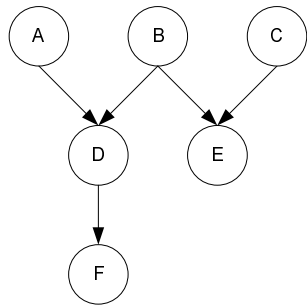
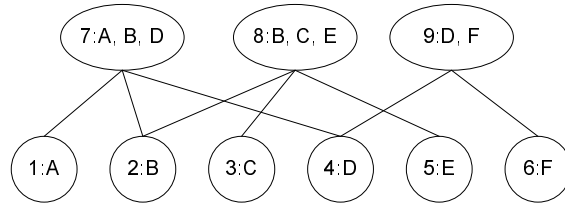


Figure 7: Example of two decision trees



(a) Bayesian Network



(b) Bethe Cluster Graph

Figure 8: Example of Bethe Cluster Graph

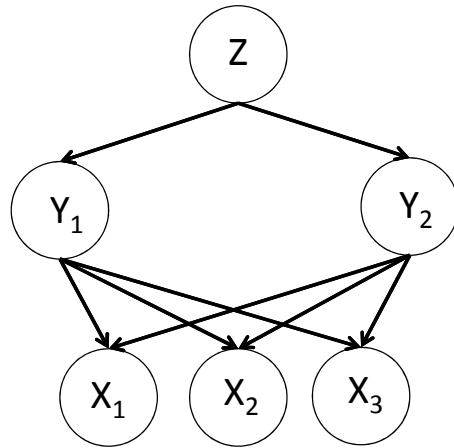


Figure 9: Simple system

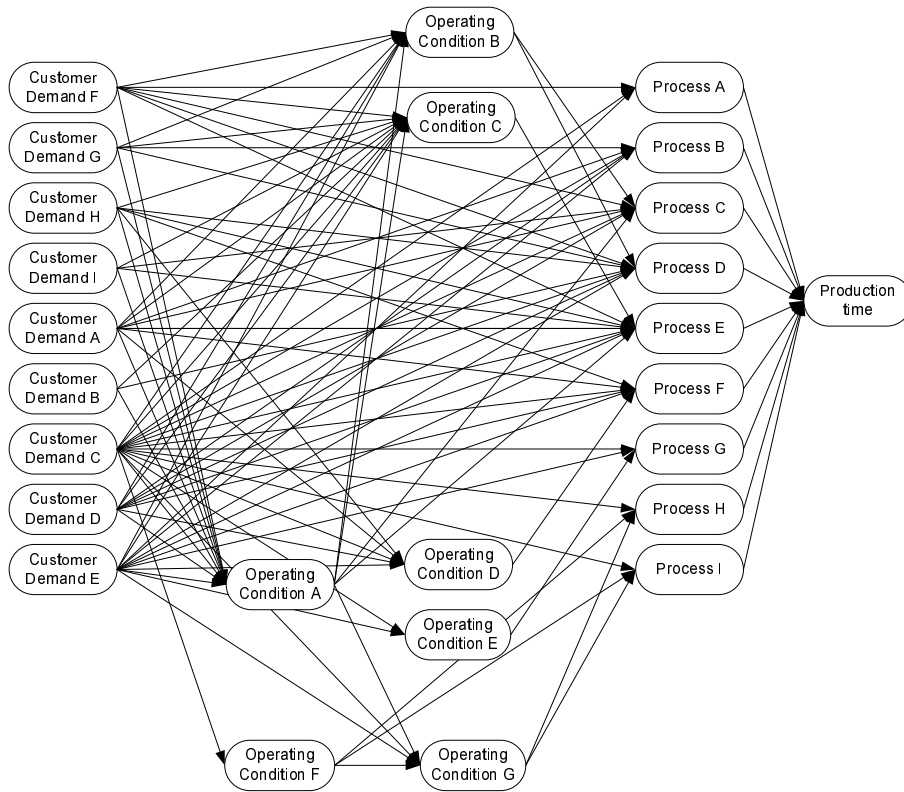


Figure 10: Identified first type of Bayesian network for steel production process

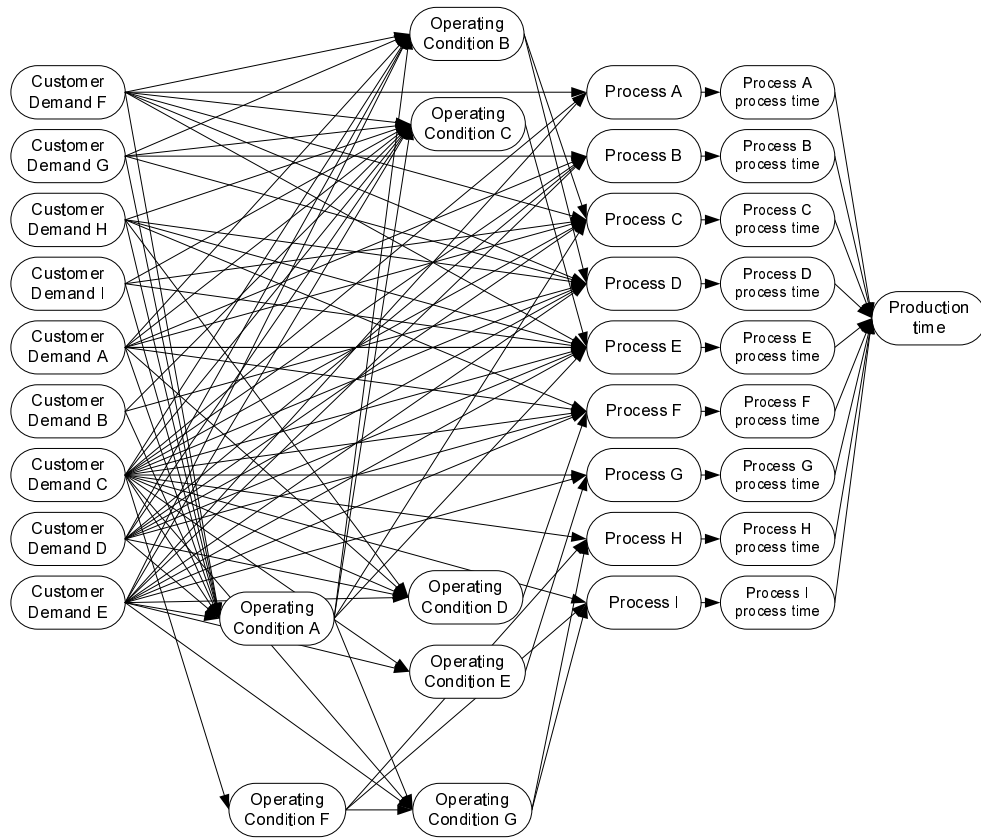


Figure 11: Identified second type of Bayesian network for steel production process

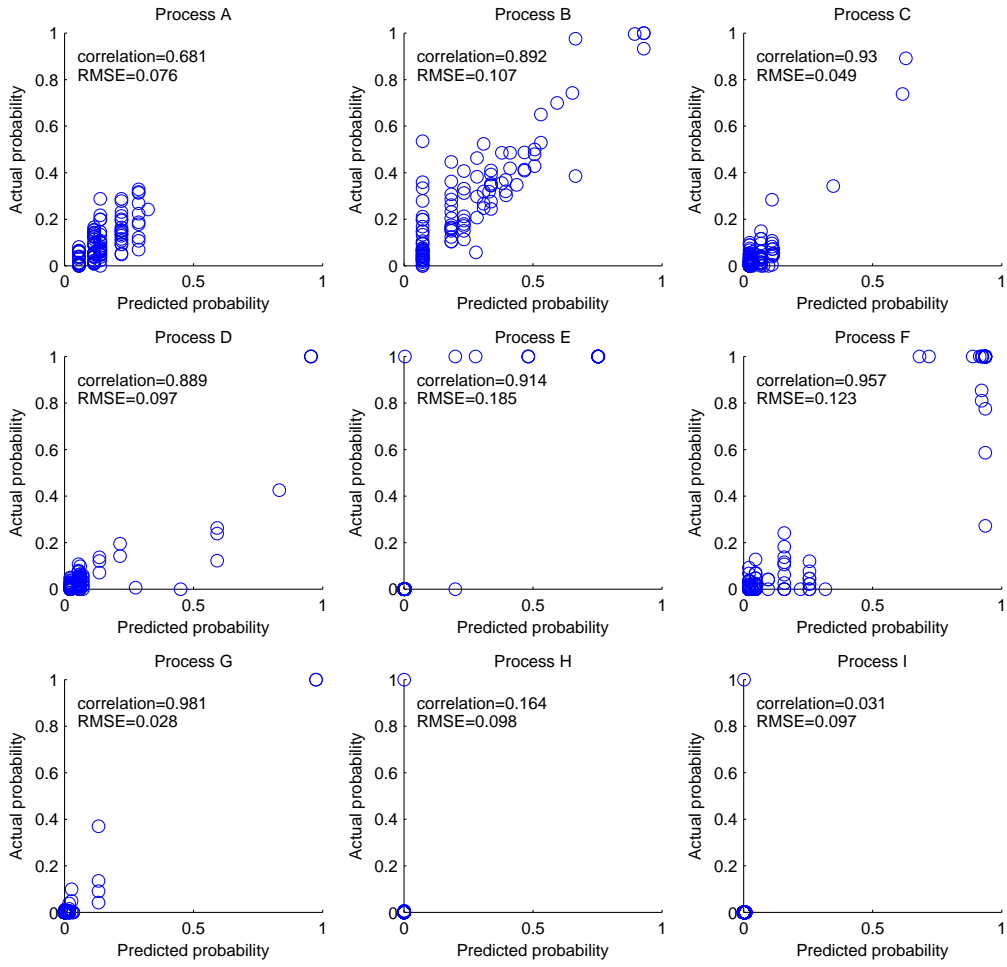


Figure 12: Scatter plots of probability of production loads given customer demands in case 1

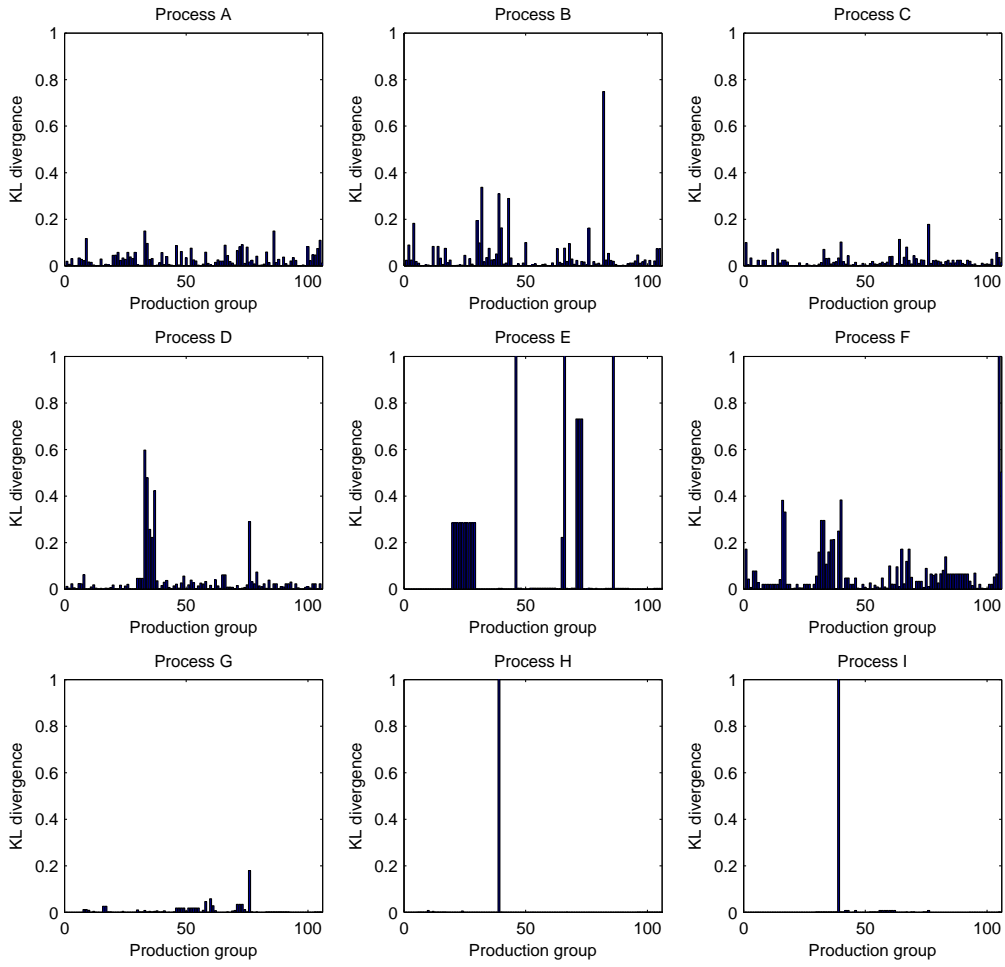


Figure 13: KL divergences between the predicted and true probability distributions of production loads given customer demands in case 1

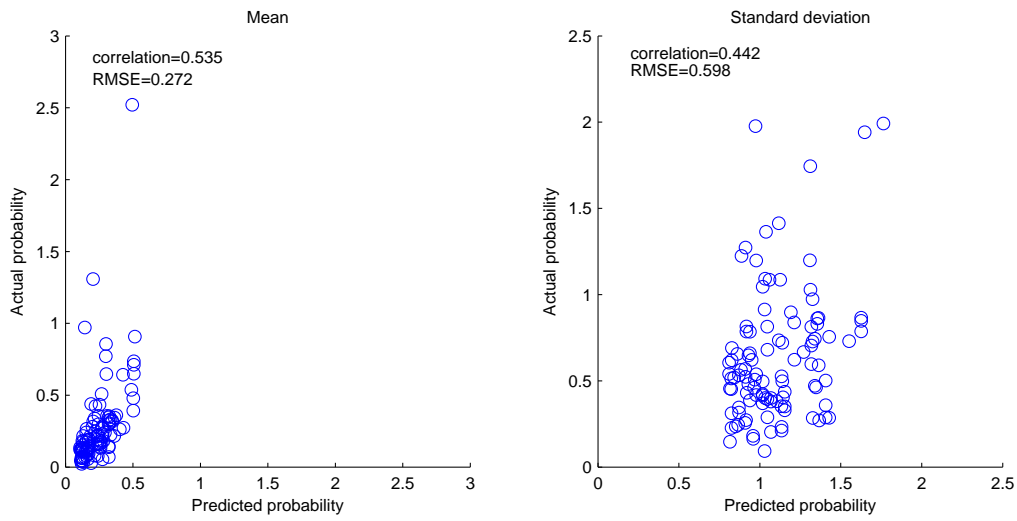


Figure 14: Scatter plots of mean and standard deviation of production time given customer demands using the first Bayesian network in case 1

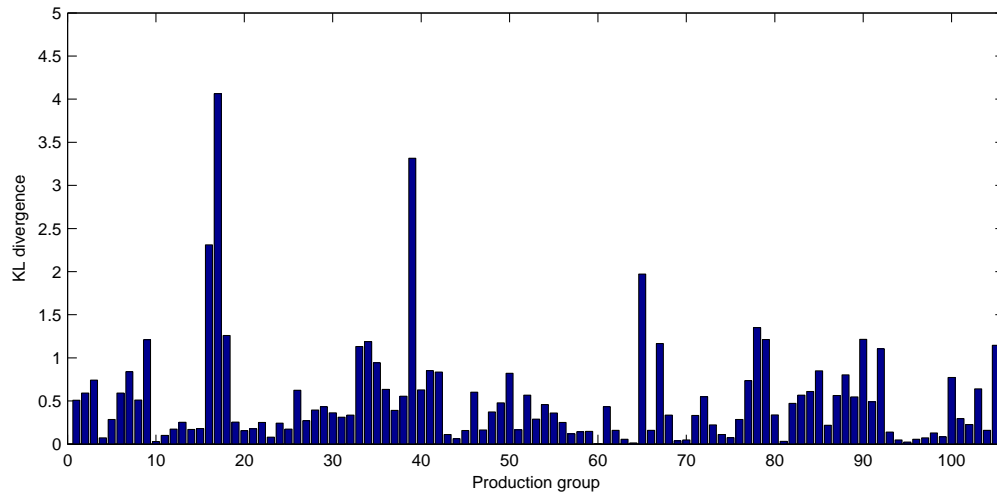


Figure 15: KL divergences between the predicted and true probability distributions of production time given customer demands using the first Bayesian network in case 1

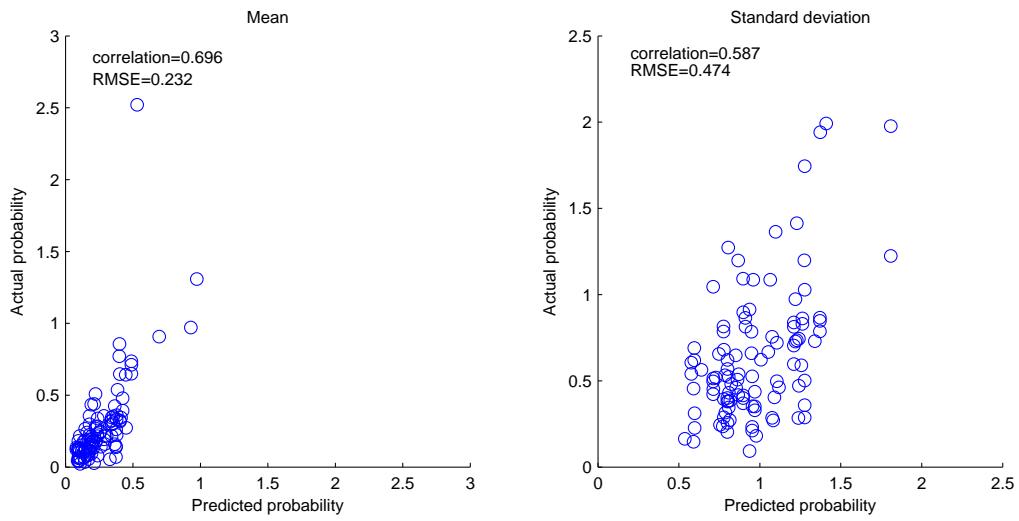


Figure 16: Scatter plots of mean and standard deviation of production time given customer demands using the second Bayesian network in case 1

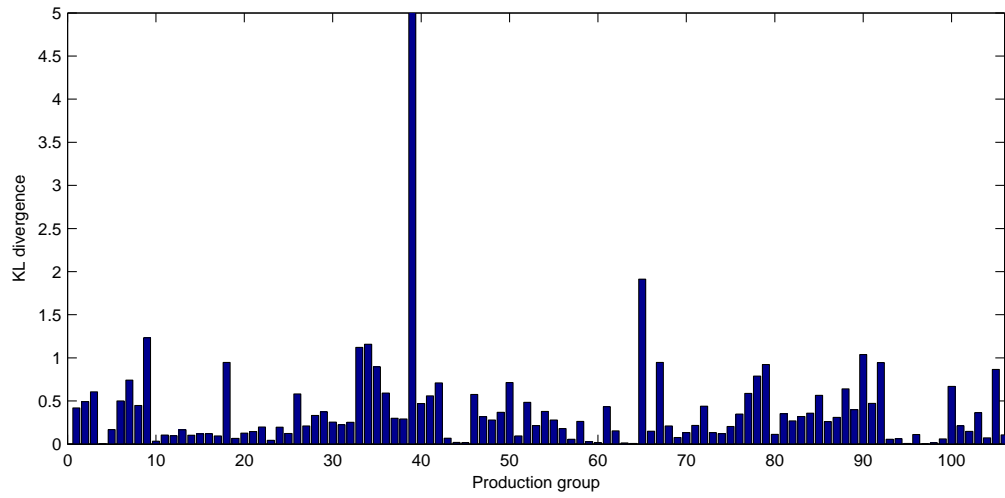


Figure 17: KL divergences between the predicted and true probability distributions of production time given customer demands using the second Bayesian network in case 1

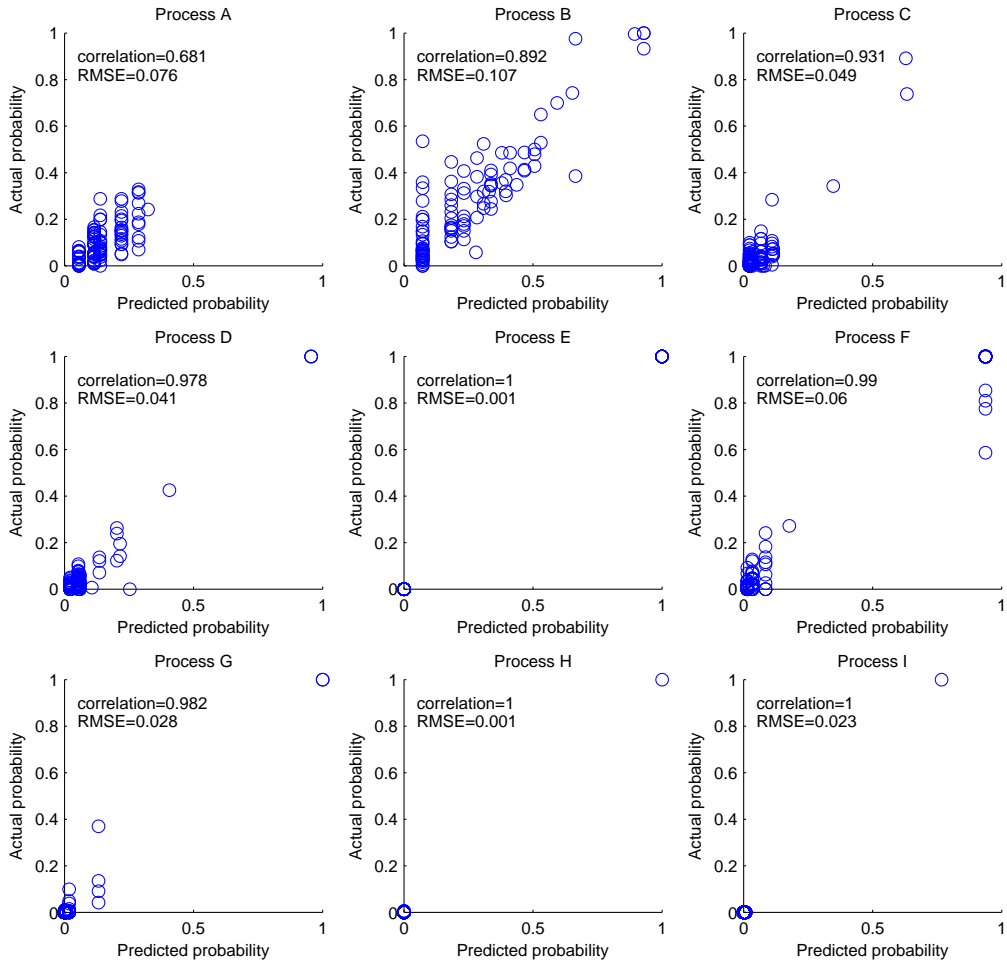


Figure 18: Scatter plots of probability of production loads given customer demands and operating conditions in case 2

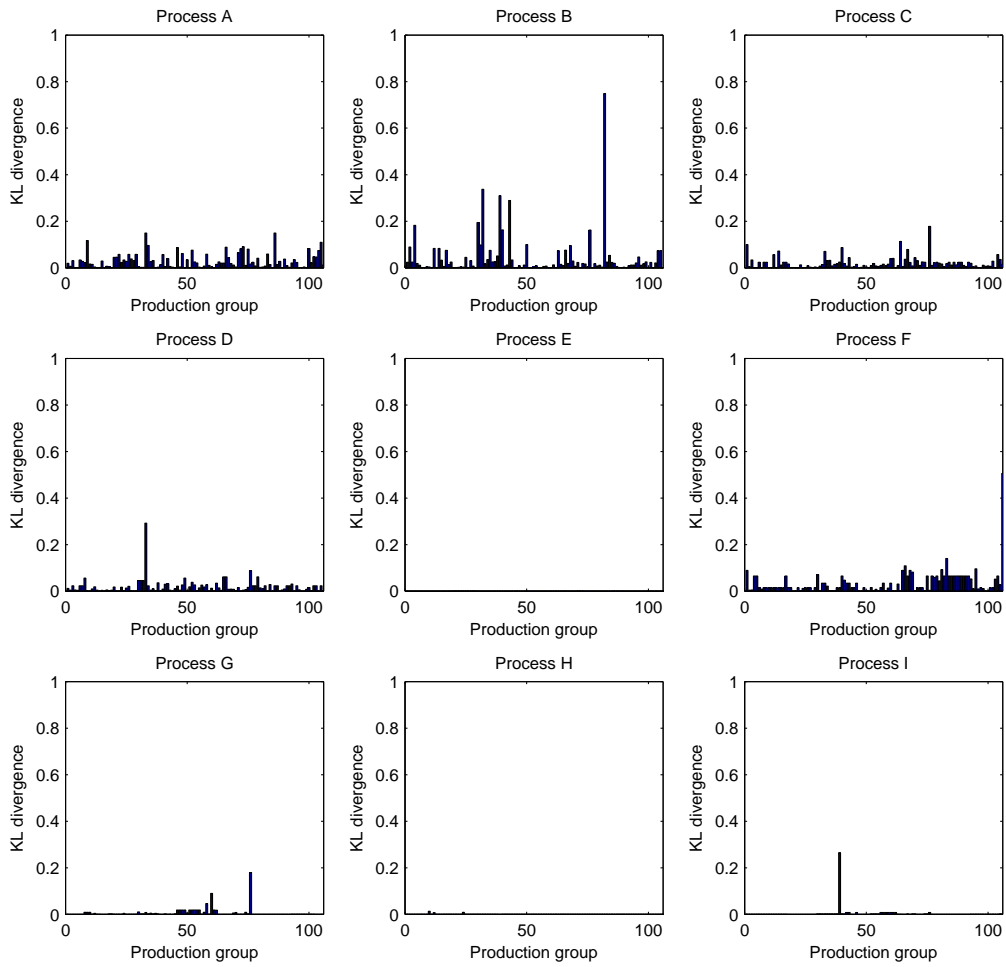


Figure 19: KL divergences between the predicted and true probability distributions of production loads given customer demands and operating conditions in case 2

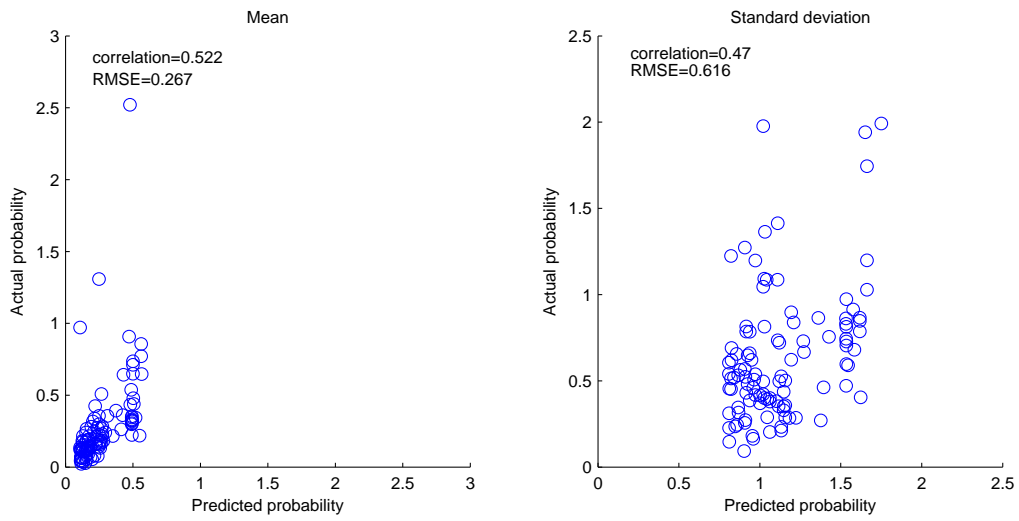


Figure 20: Scatter plots of mean and standard deviation of production time given customer demands and operating conditions using the first Bayesian network in case 2

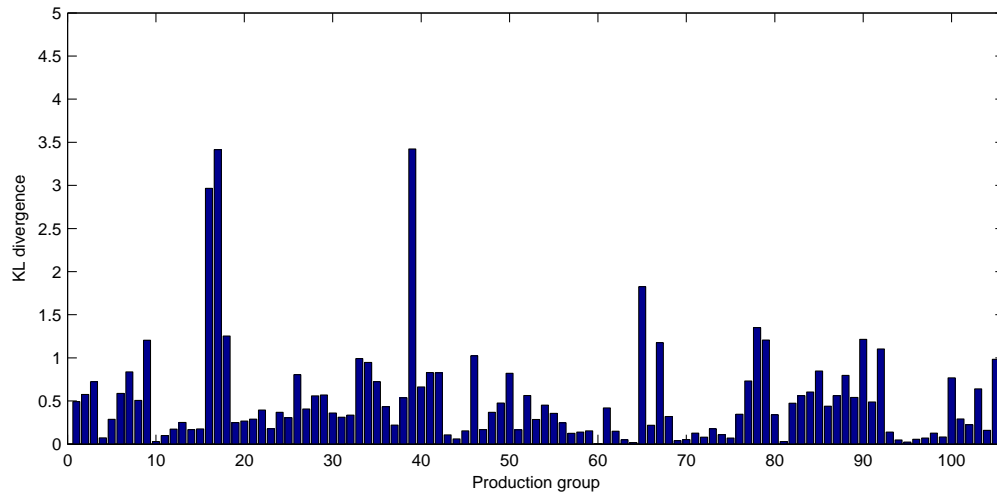


Figure 21: KL divergences between the predicted and true probability distributions of production time given customer demands and operating conditions using the first Bayesian network in case 2

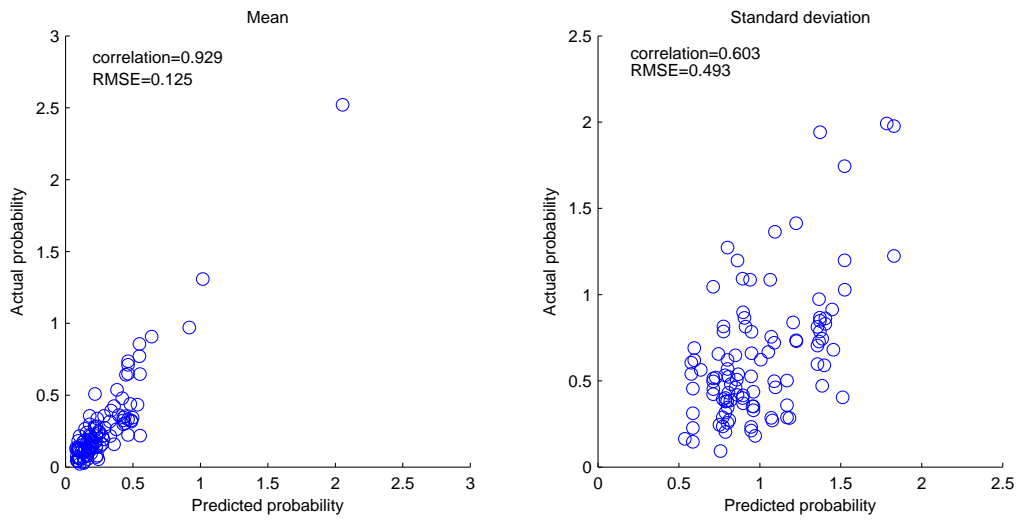


Figure 22: Scatter plots of mean and standard deviation of production time given customer demands and operating conditions using the second Bayesian network in case 2

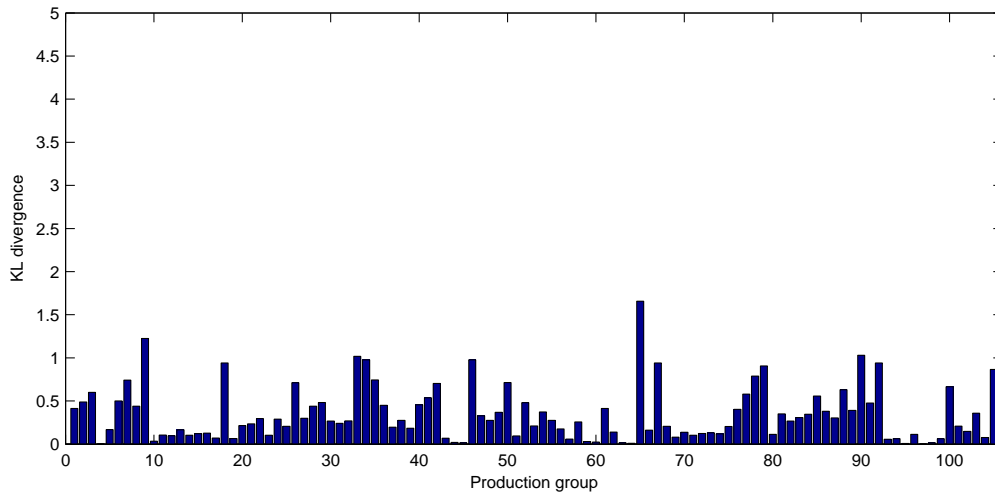


Figure 23: KL divergences between the predicted and true probability distributions of production time given customer demands and operating conditions using the second Bayesian network in case 2

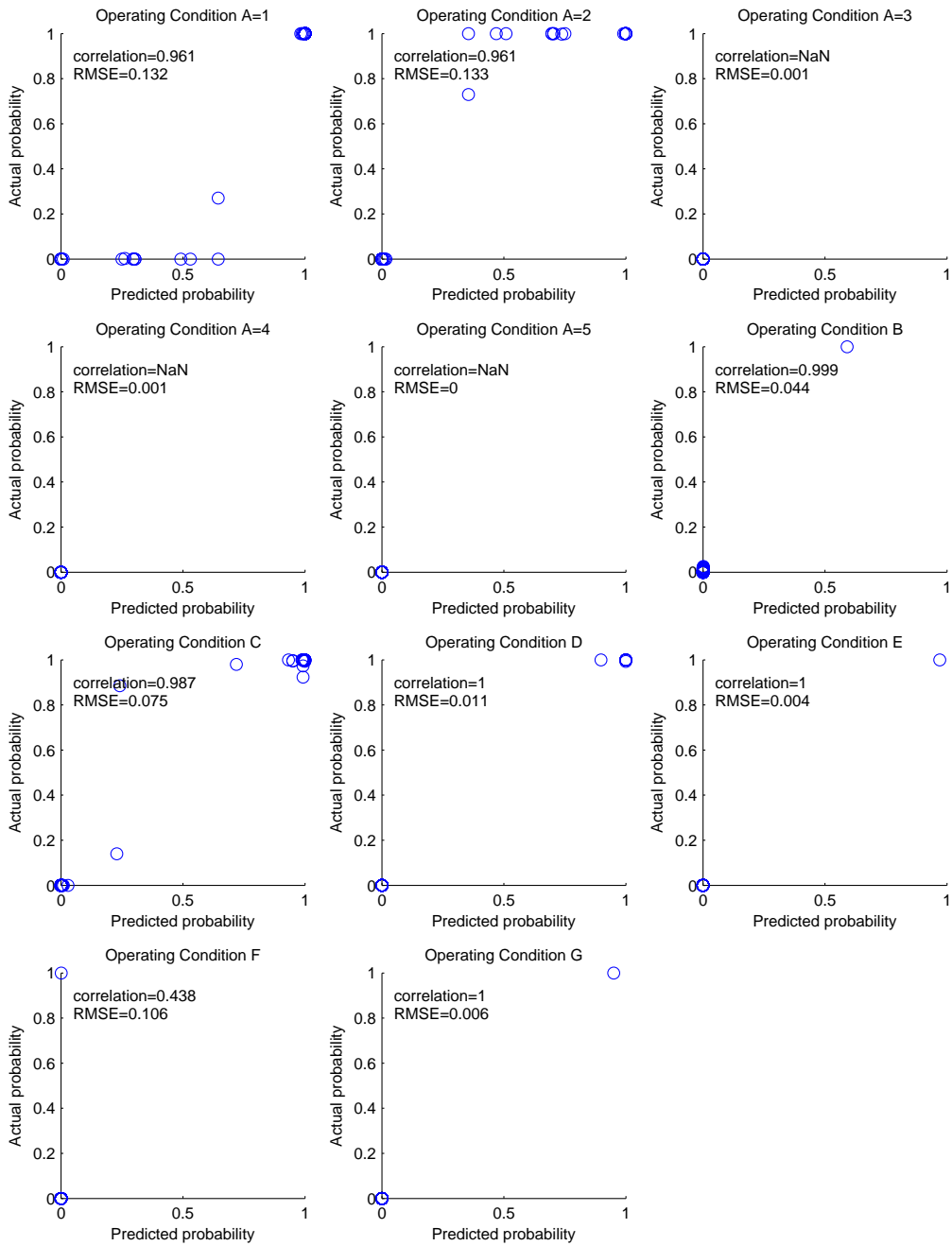


Figure 24: Scatter plots of probability of each cardinality of operating conditions given customer demands and production loads in case 3

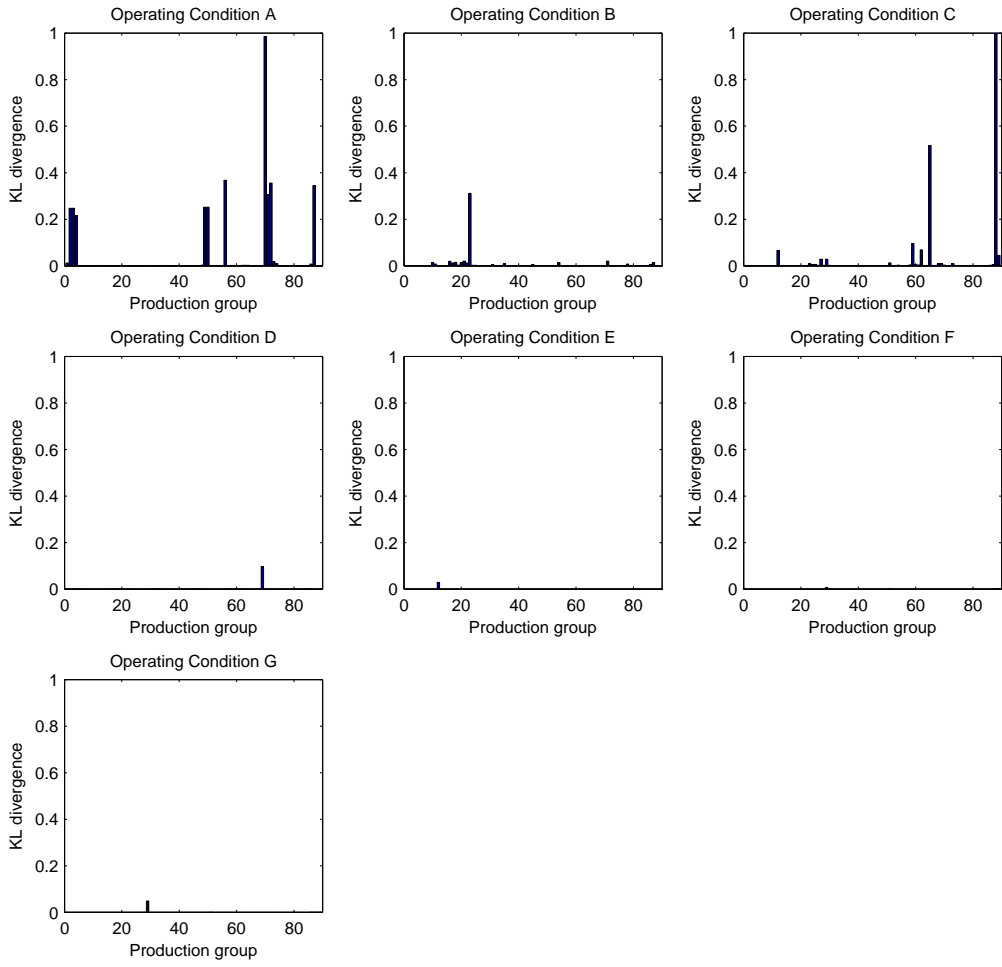


Figure 25: KL divergences between the predicted and true probability distributions of operating conditions given customer demands and production loads in case 3