Article Pre-Print

Inference in Hybrid Bayesian Networks with Large Discrete and Continuous Domains

Junichi Mori[a,1], Vladimir Mahalec [b,2,*]

[a] *Kimitsu City, Chiba Pref., Japan 299-1141*
[b] *McMaster University, Hamilton, Ontario, Canada L8S 4L7*

Abstract

Inference in Bayesian networks with large domain of discrete variables requires significant computational effort. In order to reduce the computational effort, current approaches often assume that discrete variables have some bounded number of values or are represented at an appropriate size of clusters. In this paper, we introduce decision-tree structured conditional probability representations that can efficiently handle a large domain of discrete and continuous variables. These representations can partition the large number of values into some reasonable number of clusters and lead to more robust parameter estimation. Very rapid computation and ability to treat both discrete and continuous variables are accomplished via modified belief propagation algorithm. Being able to compute various types of reasoning from a single Bayesian network eliminates development and maintenance issues associated with the use of distinct models for different types of reasoning. Application to real-world steel production process data is presented.

Keywords: large domain of discrete and continuous variables, decision tree, hybrid Bayesian network, belief propagation algorithm, context-specific independence

*Corresponding author
Email addresses: mori.nn4.junichi@jp.nssmc.com (Junichi Mori), mahalec@mcmaster.ca (Vladimir Mahalec)
[1] Nippon Steel & Sumitomo Metal Corporation, Kimitsu Works, Department of Computer System
[2] McMaster University, Department of Chemical Engineering

1. Introduction

This work has been motivated by the need to optimize production of steel plates manufacturing. Steel plates manufacturing is a complex, multi-stage process. A manufacturing plant often produces several thousands of different steel plate SKUs. Each stage of the manufacturing is not a fully deterministic process. Sometimes there may be defects at some stage, which are then corrected by modifying the manufacturing process. In other instances, a customer may order a new steel plate, something that has not been manufactured yet. Due to these circumstances, the exact times required to produce any specific SKU is not known. Production planning and scheduling models require that we estimate the production times for each grade of steel plates. Such estimate can be made from a Bayesian network representing the manufacturing plant and probabilities of processing a steel plate at each stage of the manufacturing. Due to the complexity of the steel manufacturing plant, it is not possible to use a first-principle model of the plant to construct such Bayesian network. In this paper we use Bayesian statistics to construct the most likely Bayesian network for such complex manufacturing process. Having constructed Bayesian network, we then proceed to estimate most likely production times for each grade of steel plates. The challenging nature of the problem is magnified by the large number of different grades of steel plates. This paper presents new inference algorithm in Bayesian network with large domain discrete variables to enable us to:

1. Estimate the probability distributions of production production time from historical data with large domain discrete variables and continuous variables.

2. Deal with unobservable (unavailable) variables such that we have a single model and avoid multiple models that meet with specific problems.

In this section we review prior related works and discuss the limitations with respect the size of the problem and complexity of computation.

Let us first review prior related works and discuss the capabilities of the proposed methods with respect the size of the problem and complexity of computation.

Probabilistic graphical models are popular for representing conditional independencies among random variables under system uncertainty. Such models are comprised of nodes representing random variables and the links

between the nodes which express probabilistic relationships among the corresponding random variables. Two major classes of graphical models are *Bayesian networks* and *Markov random fields*. Bayesian networks are also called *directed graphical models* since the links of the graphs represent direct dependence among the variables and are described by arrows between links. Markov random fields are also called *undirected graphical models* since they provide a simple definition of independence among random variables and do not have a particular directionality indicated by arrows (Pearl, 1988; Bishop, 2006). Both graphical models are popular in the machine learning community and have been applied to various fields including medical diagnostics, speech recognition, gene modeling, cancer classification, target tracking, sensor validation, and reliability analysis.

In particular, Bayesian network has been widely used for systems including many uncertainties. For instance, scenario analysis under changing conditions is implemented by means of Bayesian inference techniques, since Bayesian network performs well in uncertainty environment (Buyukozkan et al., 2015; Cai et al., 2011). Bayesian network is also applied to predicting the risk of software development or maintenance projects, because it is suitable for representing the knowledge of experts under uncertainty of conditions (Perkusich and Soares, 2014; Melo and Sanchez, 2008). As these applications indicate, Bayesian network is a powerful tool for knowledge representation and reasoning under uncertainties since it can visually represents the probabilistic relationships among measured and unmeasured variables.

Each node in a Bayesian network is associated with conditional probability distributions (CPD). The most common representations of CPDs are conditional probability tables (CPTs), which specify marginal probability distributions for each combination of values of its discrete parent nodes. The number of parameters required to represent CPTs grows exponentially both with the number of discrete variables and with the cardinality of discrete variables. In order to reduce the number of parameters, context-specific independence representations have been proposed (Boutilier et al., 1996). Furthermore, an efficient inference algorithm that exploits context-specified independence has also been proposed (Poole and Zhang, 2003). As for identification of parameters of context-specific independence, learning methods such as tree-structured CPTs (Friedman and Goldszmidt, 1996) and graph-structured CPTs (Chickering et al., 1997) have been developed. However, since learning structured CPTs is NP-hard problems, all of these methods assume that all discrete variables have a bounded number of values or that

they are already grouped at an appropriate level of domain size. In the real world problem, discrete variables often have large domains and the task of grouping discrete values requires expert knowledge that enables us to identify a reasonable set of groups that well distinguish the values of discrete variables. In order to group the discrete values in a Bayesian network learning, $attribute - value\ hierarchies$ (AVHs) which capture meaningful groupings of values in a particular domain are integrated with the tree-structured CPTs (DesJardins and Rathod, 2008). However, if large domain discrete variables do not contain hierarchal structures, AVHs cannot capture the useful abstracts of values in that domain. In addition, this model cannot handle the continuous variables without discretizing them. The authors, DesJardins and Rathod (2008) also do not apply AVH-derived CPTs to inference in Bayesian networks.

Sharma and Poole (2003) have proposed an inference method for Bayesian Networks containing CPTs which are represented as decision trees. The inference algorithm is based on variable elimination (VE) algorithm; the authors introduced operations for decision trees computations, namely multiplying factors and summing out variable from a factor. However, because the computational complexity of the exact inference algorithm such as VE grows exponentially with the size of the network, this method may not be appropriate for Bayesian networks in real world. In addition, computational cost of reconstructing decision trees as required to compute multiplication and marginalization of factors makes this method too expensive to apply to large decision trees. An alternative approach is to employ algebraic decision diagrams (ADDs) for the purpose of inference in Bayesian network with large domain discrete variables. For instance, ADDs have been used to represent factors and their multiplying and summing-out operations have been proposed (Chavira and Darwiche, 2007). In addition, structured message passing has been proposed to utilize powerful approximate inference algorithms such as cluster-graph Belief propagation (Gogate and Domingos, 2013). In the worst case, ADDs have the same space complexity as CPTs. To make matters worse, the factor operations of multiplication and summing-out are polynomial in time. It shold be noted that all the above methods have not considered an application of 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 inference is the conditional linear Gaussian (CLG) model (Lauritzen,

1992; Lauritzen and Jensen, 2001). The corresponding graphical 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. Since there is no exact inference algorithm for such model, the inference is typically carried out by means of approximate inference such as Mote Carlo method (Koller et al., 1999). One problem with the Monte Carlo method is that the convergence can be quite slow especially when we deal with the hybrid networks with large domain of discrete variables. An alternative is to discretize all continuous variables in a network and treat them as if they are discrete (Kozlov and Koller, 1997). However, since the number of variables in intermediate factors can be quite large, it is typically impossible to discretize the continuous variables as finely as required to obtain reasonable solutions. This discretization leads to a trade-off between accuracy of the approximation and the cost of computation.

An alternative approach is the mixture of truncated exponential (MTE) model to represent the hybrid Bayesian networks (Moral et al., 2001). 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 algorithm can be used. Parameter estimation and propagation algorithms for MTE models have been extensively studied (Rumi and Salmeron, 2007; Cobb et al., 2007). This method is also limited to relatively modest size models, since the number of regression coefficients in exponential functions linearly grows in the domain size of discrete variables. Hence, MTE model may not work well in the large Bayesian networks that contain large domain of discrete variables. Extended probability trees for probabilistic graphical models are also proposed (Cano et al., 2014). These trees allow the representation of multiplicative and additive factorization. However, similar idea as MTE is employed to operate with discrete distributions and thus this algorithm is also limited to modest size models.

In this paper we present a new method for constructing most likely structure of the Bayesian network representing a complex manufacturing process, e.g. manufacturing of steel plates. Such networks contain a large number of discrete variables and also contain continuous variables parent nodes which have discrete variables children nodes. An application to a steel plate manufacturing process, which produces a large number of distinct steel plates and also has uncertain production times at different manufacturing steps, demonstrates that the proposed method can successfully compute inferences

for very large hybrid Bayesian networks.

In order to learn the tree structured CPTs, scores such as Bayesian scores are often used as objective functions. However, greedy hill climbing approach cannot be used since it can easily get stuck in a local minimum at the early stage. Some kinds of approach to avoid local minimum as much as possible are proposed, but they are computationally expensive. Therefore, we employ decision trees algorithm (Breiman et al., 1984) based on classification trees in order to learn the tree structured CPTs. Classification trees predict the dependent variables following decisions in the tree from the root node down to the leaf node. Since the classification trees group the values to capture important distinctions of continuous or discrete variables, this method can be used to construct the context-specific CPTs in the hybrid Bayesian networks. The classification tree classifies discrete variables into a small number of subsets so that the values of continuous or discrete child nodes can be distinguished well. If Bayesian networks include continuous parent nodes with discrete child nodes, 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 cost is relatively small. Consequently, the intermediate factors can be described compactly using a simple parametric representation called the canonical form.

We also propose the decision-tree structured CPT based inference algorithm in Bayesian network, which employs belief propagation algorithm to deal with hybrid networks with large domain discrete variables. In order for multiplying and marginalizing factors during belief propagation, various types of operations to dynamically construct CPTs are proposed. In order to carry out other types of inferences, such as causal, diagnostic, intercausal and mixed reasoning, we employ the loopy belief propagation in the decision-tree structured CPT based Bayesian networks.

The organization of this article is as follows. Section 2 proposes the decision-tree structured CPTs for hybrid Bayesian networks with large domain discrete variables. Section 3 describes the detailed inference algorithm including operations of product and marginalization of factors. The presented method is applied to the steel production processes data in section 4. Finally, the conclusions are presented in Section 5.

2. Decision-Tree Structured Conditional Probability Table

A hybrid Bayesian network represents a probability distribution over random variables in which each node is either discrete or continuous. $\mathcal{X} = \Gamma \cup \Delta$ represents sets of random variables ,where Γ denotes the continuous variables and Δ represents the discrete variables. We denote random discrete variables by upper case letters from the beginning of the alphabet (e.g. A, B, A_1) while continuous variables are represented by upper case letters near the end ($e.g. X, Y, X_1$). Their actual values are represented by the lower case letters (e.g. x). We denote sets of variables by bold upper case letters (e.g. $\mathbf{X}$) and the assignments of those sets by the bold lower case letters (e.g. $\mathbf{x}$). $\mathrm{Val}(X)$ is used to represent the set of values that a random variable X can take.

The most widely used representation of hybrid Bayesian networks is the conditional linear Gaussian (CLG) model. Let $X \subseteq \Gamma$ be a continuous node, $A \subseteq \mathrm{pa}(X) \cap \Delta$ be its discrete parent nodes and $Y_1, \ldots, Y_k \subseteq \mathrm{pa}(X) \cap \Gamma$ be its continuous parent nodes where $\mathrm{pa}(X)$ denotes a state of parent nodes of X. X has a Gaussian distribution and the mean of the distribution is computed as a linear combination of the state of its parent nodes $\mathrm{pa}(X)$ for every instantiation $a \in \mathrm{Val}(A)$ as follows:

$$p(X|\mathrm{pa}(X)) = P\left(X \,|\mathrm{pa}(X); \Theta_a\right) = \mathcal{N}\left(X \left| \sum_j^k w_{a,j} y_j + b_a, \sigma_a^2 \right.\right) \tag{1}$$

where $w_{a,j}$ and b_a are parameters controlling the mean, σ_a represents 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 of instantiation a. Since this representation is not Gaussian but a conditional distribution, canonical form is used as a more general representation. In the canonical table representation, a factor $C(\mathbf{X}; K_a, h_a, g_a)$ which is a function over a set of variables is defined as (Lauritzen and Wermuth, 1989):

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

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. 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 a canonical form $\exp(g_a)$ remaines for each instantiation a.

Unfortunately, the canonical table representation cannot represent the dependence of a discrete child node with continuous parent nodes. The simplest approach to resolve this issue is to discretize the variables of continuous parent nodes. As discussed in the introduction section, there is a trade-off between the accuracy of the approximation and the domain size of discretized variables. However, the decision-tree structured CPDs proposed in this paper can handle unbounded number of values in discrete variables and we can ignore this issue.

In order to compute the parameters of canonical table, one first needs to define the instantiations of the set of discrete variables. The simplest way is to use conditional probability tables. However, the number of parameters required to describe a Bayesian network increases exponentially with the domain size. In the case when a discrete node A has discrete parent nodes $\mathrm{pa}(A)$, the number of conditional probabilites is $|A| \times |\mathrm{pa}(A)|$ and it is impossible to specify all the parameters of the large domain of discrete variables. This is the case particularly when we discretize very finely the continuous parent nodes in order to get an accurate approximation, since the domain size of the discretized variables becomes large. To exploit conditional independence that holds only in certain contexts in the Bayesian networks, several works about structured CPTs have been done to capture this independence (Boutilier et al., 1996; Poole and Zhang, 2003). However, since learning structured CPTs is an NP-hard problem, all of these methods assume either that all discrete variables have a bounded number of values or that they are already grouped at an appropriate level of domain size.

To overcome these limitations of the existing methods, we employ decision tree algorithms such as classification trees and regression trees for learning structured CPTs. Classification trees give the prediction of the dependent variables by following decisions in the tree from the root node down to the leaf node. The classification trees can predict the discrete and continuous variables and the trees are identified by choosing a split based on the some statistical measures such as Gini impurity for the classification trees and mean square error for the regression trees.

Let us first consider a factor that contains both continuous nodes $\mathbf{X}$ and discrete nodes $\mathbf{D}$ with continuous child node Y as shown in Fig. 1. Continuous variables are shown as rectangles while discrete variables are ovals. The classification tree f classifies the discrete variable set $\mathbf{D}$ into a

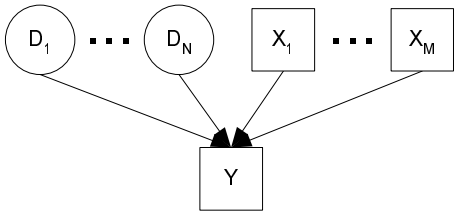


Figure 1: Bayesian network that contains continuous child node

small number of subsets $A = \{a_1, a_2, \ldots, a_L\}$ as $a = f(\mathbf{d})$ where L is the number of the leaves in the decision tree f. An example of the constructed decision tree is given in Fig. 2. For instance, the decision tree indicates that if

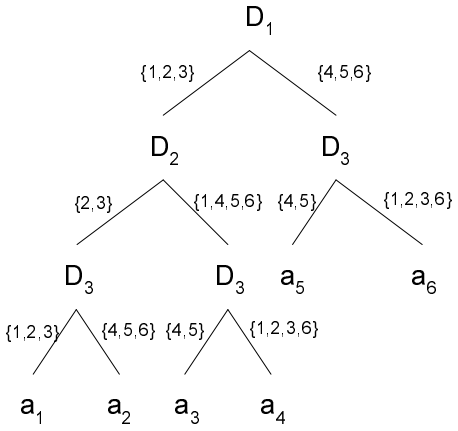


Figure 2: Decision tree of a factor that contains continuous child node

$D_1 \in \{1, 2, 3\}$, $D_2 \in \{2, 3\}$ and $D_3 \in \{1, 2, 3\}$, then the corresponding records are assigned the instantiation a_1. Since the domain size of each discrete variable is 6, the number of parameters required to describe the CPT of discrete variables $\mathbf{D}$ is $|\mathrm{Val}(D_1)| \times |\mathrm{Val}(D_2)| \times |\mathrm{Val}(D_3)| = 216$. On the other hand, the decision tree representation requires only $|\mathrm{Val}(A)| = 6$ parameters to describe the behavior of Y, instead of 216 in the CPT representation. Therefore, the decision-tree representation requires smaller training data to learn the conditional probability distribution and leads to much more robust estimation than the traditional CPT representation.

After giving instantiation $a_i \in A$ to all records in the training data,

parameters of the canonical table representation can be computed as follows:

$$\begin{aligned} K_{a_i} &= \Sigma_i^{-1} \\ \mathbf{h}_{a_i} &= \Sigma_i^{-1}\mu_i \\ g_{a_i} &= -\tfrac{1}{2}\mu_i^T {\Sigma_i}^{-1}\mu_i - \log\left((2\pi)^{n/2}|\Sigma_i|^{1/2}\right) \end{aligned} \tag{3}$$

with

$$\Sigma_i = \mathrm{cov}\left[\mathbf{Z}_i\right] \tag{4}$$

$$\mu_i = \mathrm{mean}\left[\mathbf{Z}_i\right] \tag{5}$$

where $\mathbf{Z} = \mathbf{X} \cup Y$, $\mathbf{Z}_i = \{z : f(\mathbf{d}) = a_i\}$ and $\mathrm{cov}\left[\mathbf{Z}_i\right]$ and $\mathrm{mean}\left[\mathbf{Z}_i\right]$ are the covariance matrix and mean vector of $\mathbf{Z}_i$ respectively. Tree structured CPTs induce refined independence that holds only in certain contexts, which cannot be achieved by the table representation. In the example shown in Fig. 2, if the evidence $D_1 = 4$,5 or 6 is given, we immediately know that the continuous variables $\mathbf{Z} = \mathbf{X} \cup Y$ follow the distribution either $C(\mathbf{Z}; K_{a_5}, \mathbf{h}_{a_5}, g_{a_5})$ or $C(\mathbf{Z}; K_{a_6}, \mathbf{h}_{a_6}, g_{a_6})$, regardless of the value of D_2. Thus, we can conclude that D_2 and Y are contextually independent given the evidence $D_1 = \{4, 5, 6\}$ denoted $(D_2 \perp_c Y | D_1 = \{4, 5, 6\})$, which is described as follows:

$$P\left(D_2 | Y, D_1 = \{4, 5, 6\}\right) = P\left(D_2 | D_1 = \{4, 5, 6\}\right) \quad . \tag{6}$$

Next, let us consider the case when a factor contains discrete nodes $\mathbf{D}$ with discrete child node B as shown in Fig. 3. The classification tree classi-

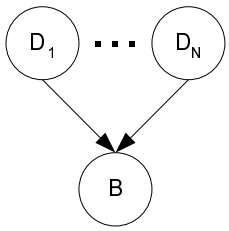


Figure 3: Bayesian network that contains discrete child node

fies discrete variables $\mathbf{D}$ into a small number of subsets $A = \{a_1, a_2, \ldots, a_L\}$ as $a = f(\mathbf{d})$ with L being the number of leaves so that the values of a discrete child node B can be categorized well. The decision tree can categorize well the discrete parent variables $\mathbf{D}$, but a discrete child variable B

(which is used as a dependent variable in the classification tree algorithm) is not included in the nodes of the decision tree. Therefore, the obtained decision tree classifies only the discrete parent variables $\mathbf{D}$ only and excludes a discrete child variable B from its branching nodes. In order to use all discrete variables (including a discrete child variable) as the brunching nodes, the discrete child variable is placed at all leaf nodes to continue categorizing discrete variables by using their values. Therefore, all values of a discrete child variable $b_1 \dots b_L \in B$ are used as categorical splits and thus the number of branches at each leaf node in the original tree is same as the domain size of $\mathrm{Val}(B) = L$. As a result, the total number of instantiations assigned in the final decision tree is $\mathrm{Val}(B)$ times as many as the number of leaves in the original decision tree. Fig. 4 shows the example of a decision tree where a factor contains a discrete child node. The

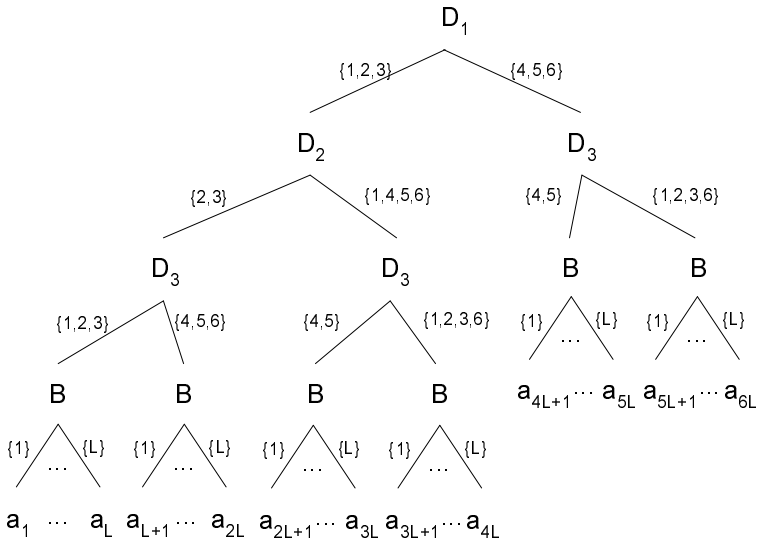


Figure 4: Decision tree of a factor that contains discrete child node

decision tree indicates that if $D_1 \in \{1, 2, 3\}$, $D_2 \in \{2, 3\}$, $D_3 \in \{1, 2, 3\}$ and $B \in \{1\}$, then the corresponding records are assigned the instantiation a_1. The number of instantiations are $\mathrm{Val}(B) \times 6 = 6L$ rather than $|\mathrm{Val}(D_1)| \times |\mathrm{Val}(D_2)| \times |\mathrm{Val}(D_3)| \times |\mathrm{Val}(B)| = 216L$ which is the number of parameters required to describe the traditional CPT of discrete variables $\mathbf{D} \cup B$.

With assigned instantiations $a_i \in A$, parameters g_{a_i} of the canonical table representation can be computed below:

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

where $\mathbf{1}$ is a indicator function, N_{a_i} is the number of samples belonging to the instantiation a_i and the other parameters K_{a_i} and $\mathbf{h}_{a_i}$ are vacuous. Similar to the previous case, tree structured CPTs hold context specific independence. In the example shown in Fig. 4, if the evidence $D_1 = 4$, 5 or 6 is given, we compute each probability of Val(B), regardless of the value of D_2. Thus, it can be said that D_2 and B are contextually independent given the evidence $D_1 = \{4, 5, 6\}$ denoted $(D_2 \perp_c B | D_1 = \{4, 5, 6\})$, which is described as follows:

$$P\left(D_2 | B, D_1 = \{4, 5, 6\}\right) = P\left(D_2 | D_1 = \{4, 5, 6\}\right) \quad . \tag{8}$$

Since this implementation uses a binary tree algorithm to create decision tree, the complexity is $O(\log N)$ with N being the number of samples.

3. Inference

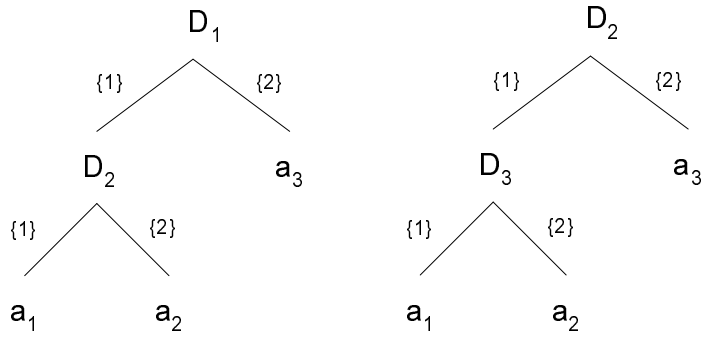
The computational task of inference in Bayesian network is to answer the *conditional probability query*, $P(\mathcal{X}|\mathbf{E} = \mathbf{e})$ where the evidence $\mathbf{E} \subseteq \mathcal{X}$ is a subset of random variables in the Bayesian network and $\mathbf{e} \in$ Val($\mathbf{E}$) is the set of values given to these variables. For causal reasoning, the sum-product algorithm is carried out from top node factors to bottom node factors. Meanwhile, other types of inference such as diagnostic, intercausal and mixed reasoning do not have monotonic logic based algorithm and the basic algorithm for exact inference for these types of reasoning in graphical models is *variable elimination* (VE). VE can be performed in a clique tree and each clique in the clique tree is the factor in VE. The computational cost of the exact inference such as clique tree algorithm exponentially grows with the width of the network and thus the exact inference algorithms are infeasible for large width networks. On the other hand, approximate inference algorithms such as loopy belief propagation (BP) algorithm are tractable for many real-world graphical models. All the above inference techniques require computation of the multiplying factors and marginalizing variables in factors. In this work, since the factors are represented by using decision-tree structured CPTs as proposed in the previous section, the operations of multiplication and marginalization are described in the first parts of this section.

3.1. Multiplying Factors and Marginalizing Over Variables in Factors

Initial potentials of factors are equal to the decision-tree structured CPTs while potentials of final and intermediate factors including messages between clusters are computed by the operations of multiplying and marginalizing factors. These operations require us to divide the large domain discrete variables in accordance with the decision-tree structure. For this purpose, in this section we describe three types of two main operations: multiplying factors and marginalizing over variables in factors.

3.1.1. Full Conditional Probability Table based Operations

The simplest approach is to convert the tree structured CPTs into conditional probability tables which can be analyzed by using standard multiplying and marginalizing techniques. We shall call this CPT *full conditional probability table* (full CPT). Let us consider the simple example of multiplying two factors described in Fig. 5. Domain size of each discrete variable is 2 ($\mathrm{Val}(D_1) = \mathrm{Val}(D_2) = \mathrm{Val}(D_3) = 2$) and the number of instantiations of each tree is 3 (a_1, a_2, a_3).



(a) Tree for Factor A (b) Tree for Factor B

Figure 5: Example of two decision trees

First, a full CPT is designed in accordance with the decision-tree structured CPTs as shown in Table 1 and using these full CPTs, a new factor C of multiplying two factors is determined as shown in Table 3. For instance, let us consider the case where $D_1 = 1, D_2 = 2, D_3 = 1$. According to the full CPTs, the first row in the full CPT for factor A shown in Table 1 and the second row in the full CPT for factor B shown in Table 2 are used for

Table 1: Full CPT for Factor A

D_1	D_2	Probability Distribution
1	1	$C(\mathbf{X}; K^A_{a_1}, \mathbf{h}^A_{a_1}, g^A_{a_1})$
2	1	$C(\mathbf{X}; K^A_{a_3}, \mathbf{h}^A_{a_3}, g^A_{a_3})$
1	2	$C(\mathbf{X}; K^A_{a_2}, \mathbf{h}^A_{a_2}, g^A_{a_2})$
2	2	$C(\mathbf{X}; K^A_{a_3}, \mathbf{h}^A_{a_3}, g^A_{a_3})$

computing a new factor $C(\mathbf{X}; K^C_{a_3}, \mathbf{h}^C_{a_3}, g^C_{a_3})$. The canonical parameters of this new factor are computed as follows:

$$\begin{aligned} K^C_{a_3} &= K^A_{a_2} + K^B_{a_3} \\ \mathbf{h}^C_{a_3} &= \mathbf{h}^A_{a_2} + \mathbf{h}^B_{a_3} \\ g^C_{a_3} &= g^A_{a_2} + g^B_{a_3}. \end{aligned} \tag{9}$$

Since the scopes of two factors are different, we need to add zero entries

Table 2: Full CPT for Factor B

D_2	D_3	Probability Distribution
1	1	$C(\mathbf{X}; K^B_{a_1}, \mathbf{h}^B_{a_1}, g^B_{a_1})$
2	1	$C(\mathbf{X}; K^B_{a_3}, \mathbf{h}^B_{a_3}, g^B_{a_3})$
1	2	$C(\mathbf{X}; K^B_{a_2}, \mathbf{h}^B_{a_2}, g^B_{a_2})$
2	2	$C(\mathbf{X}; K^B_{a_3}, \mathbf{h}^B_{a_3}, g^B_{a_3})$

to the K matrices and $\mathbf{h}$ vectors of both factors so that their scopes are consistent.

Next, let us consider summing out variables in factors. 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 Gaussian distributions. Therefore, mixtures of Gaussians model can be used to represent the factors and Monte Carlo integration are carried out to compute the marginalization (Yuan and Druzdzel, 2006). However, since the number of Gaussian components grow every time after summing out the discrete variables, we typically approximate the mixture of Gaussian distributions by collapsing them into a single Gaussian distribution. Similarly to multiplying the factors, first we convert the decision-tree structured

CPTs into full conditional probability tables. In practice since marginalization operations are needed only after multiplying operations, full conditional probability tables are already obtained when marginalization is necessary. Let us consider summing out variable D_3 in the full CPT given in Table 3. The marginal distribution over D_1 and D_2 assigns probability distribu-

Table 3: Full CPT for Factor C

D_1	D_2	D_3	Probability Distribution
1	1	1	$C(\mathbf{X}; K^C_{a_1}, \mathbf{h}^C_{a_1}, g^C_{a_1}) = C(\mathbf{X}; K^A_{a_1}, \mathbf{h}^A_{a_1}, g^A_{a_1}) \cdot C(\mathbf{X}; K^B_{a_1}, \mathbf{h}^B_{a_1}, g^B_{a_1})$
2	1	1	$C(\mathbf{X}; K^C_{a_2}, \mathbf{h}^C_{a_2}, g^C_{a_2}) = C(\mathbf{X}; K^A_{a_3}, \mathbf{h}^A_{a_3}, g^A_{a_3}) \cdot C(\mathbf{X}; K^B_{a_1}, \mathbf{h}^B_{a_1}, g^B_{a_1})$
1	2	1	$C(\mathbf{X}; K^C_{a_3}, \mathbf{h}^C_{a_3}, g^C_{a_3}) = C(\mathbf{X}; K^A_{a_2}, \mathbf{h}^A_{a_2}, g^A_{a_2}) \cdot C(\mathbf{X}; K^B_{a_3}, \mathbf{h}^B_{a_3}, g^B_{a_3})$
2	2	1	$C(\mathbf{X}; K^C_{a_4}, \mathbf{h}^C_{a_4}, g^C_{a_4}) = C(\mathbf{X}; K^A_{a_3}, \mathbf{h}^A_{a_3}, g^A_{a_3}) \cdot C(\mathbf{X}; K^B_{a_3}, \mathbf{h}^B_{a_3}, g^B_{a_3})$
1	1	2	$C(\mathbf{X}; K^C_{a_5}, \mathbf{h}^C_{a_5}, g^C_{a_5}) = C(\mathbf{X}; K^A_{a_1}, \mathbf{h}^A_{a_1}, g^A_{a_1}) \cdot C(\mathbf{X}; K^B_{a_2}, \mathbf{h}^B_{a_2}, g^B_{a_2})$
2	1	2	$C(\mathbf{X}; K^C_{a_6}, \mathbf{h}^C_{a_6}, g^C_{a_6}) = C(\mathbf{X}; K^A_{a_3}, \mathbf{h}^A_{a_3}, g^A_{a_3}) \cdot C(\mathbf{X}; K^B_{a_2}, \mathbf{h}^B_{a_2}, g^B_{a_2})$
1	2	2	$C(\mathbf{X}; K^C_{a_7}, \mathbf{h}^C_{a_7}, g^C_{a_7}) = C(\mathbf{X}; K^A_{a_2}, \mathbf{h}^A_{a_2}, g^A_{a_2}) \cdot C(\mathbf{X}; K^B_{a_3}, \mathbf{h}^B_{a_3}, g^B_{a_3})$
2	2	2	$C(\mathbf{X}; K^C_{a_8}, \mathbf{h}^C_{a_8}, g^C_{a_8}) = C(\mathbf{X}; K^A_{a_3}, \mathbf{h}^A_{a_3}, g^A_{a_3}) \cdot C(\mathbf{X}; K^B_{a_3}, \mathbf{h}^B_{a_3}, g^B_{a_3})$

tions 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:

$$P(D_1 = 1, D_2 = 1) = C(\mathbf{X}; K^C_{a_1}, \mathbf{h}^C_{a_1}, g^C_{a_1}) + C(\mathbf{X}; K^C_{a_5}, \mathbf{h}^C_{a_5}, g^C_{a_5}). \quad (10)$$

As mentioned before, however, this distribution is not a Gaussian and thus it cannot be represented in 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 defined as follows:

$$\mu_j = \sum_{i=1}^{k} p(i|j)\mu_i \quad (11)$$

$$\Sigma_j = \sum_{i=1}^{k} p(i|j)\Sigma_i + \sum_{i=1}^{k} p(i|j)(\mu_i - \mu)(\mu_i - \mu)^T \quad (12)$$

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 the canonical form $P(D_1 = 1, D_2 = 1)$ after summing out variable D_3 can be computed as follows:

$$K_{\text{new}} = \Sigma_{\text{new}}^{-1} \tag{13}$$

$$\mathbf{h}_{\text{new}} = \Sigma_{\text{new}}^{-1} \mu_{\text{new}} \tag{14}$$

where

$$\begin{aligned} \mu_{\text{new}} = {} & p(D_3 = 1|D_1 = 1, D_2 = 1) K_{a_1}^{C-1} \mathbf{h}_{a_1}^{C} \\ & + p(D_3 = 2|D_1 = 1, D_2 = 1) K_{a_5}^{C-1} \mathbf{h}_{a_5}^{C} \end{aligned} \tag{15}$$

$$\begin{aligned} \Sigma_{\text{new}} = {} & p(D_3 = 1|D_1 = 1, D_2 = 1) K_{a_1}^{C-1} + p(D_3 = 2|D_1 = 1, D_2 = 1) K_{a_5}^{C-1} \\ & + p(D_3 = 1|D_1 = 1, D_2 = 1)(K_{a_1}^{C-1} \mathbf{h}_{a_1}^{C} - \mu_{\text{new}})(K_{a_1}^{C-1} \mathbf{h}_{a_1}^{C} - \mu_{\text{new}})^T \\ & + p(D_3 = 2|D_1 = 1, D_2 = 1)(K_{a_5}^{C-1} \mathbf{h}_{a_5}^{C} - \mu_{\text{new}})(K_{a_5}^{C-1} \mathbf{h}_{a_5}^{C} - \mu_{\text{new}})^T. \end{aligned} \tag{16}$$

After summing out discrete variables, the operation of continuous marginalization is carried out if we need to integrate the continuous variables $\mathbf{Y}$. We assume that the 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 after summing out discrete variables where

$$K_{a_i} = \begin{bmatrix} K_{\mathbf{XX}} & K_{\mathbf{XY}} \\ K_{\mathbf{YX}} & K_{\mathbf{YY}} \end{bmatrix} \tag{17}$$

$$\mathbf{h}_{a_i} = \begin{pmatrix} \mathbf{h}_{\mathbf{X}} \\ \mathbf{h}_{\mathbf{Y}} \end{pmatrix}. \tag{18}$$

If $K_{\mathbf{YY}}$ is positive definite, then the integral over the variables $\mathbf{Y}$ becomes a canonical form $C(\mathbf{X}; K_{a_i}^{\prime C}, \mathbf{h}_{a_i}^{\prime C}, g_{a_i}^{\prime C})$ computed as (Lauritzen, 1992):

$$\begin{aligned} K_{a_i}^{\prime C} &= K_{\mathbf{XX}} - K_{\mathbf{XY}} K_{\mathbf{YY}}^{-1} K_{\mathbf{YX}} \\ \mathbf{h}_{a_i}^{\prime C} &= \mathbf{h}_{\mathbf{X}} - K_{\mathbf{XY}} K_{\mathbf{YY}}^{-1} \mathbf{h}_{\mathbf{Y}} \\ g_{a_i}^{\prime C} &= g + \tfrac{1}{2} \left(\log|2\pi K_{\mathbf{YY}}^{-1}| + \mathbf{h}_{\mathbf{Y}}^{T} K_{\mathbf{YY}}^{-1} \mathbf{h}_{\mathbf{Y}} \right) \quad . \end{aligned} \tag{19}$$

Above methodology enables accurate computation of multiplication and marginalization of factors using full CPTs. This method is easy to carry out and computationally it is very fast. However, to describe the full CPT of a

factor, the number of required rows grows exponentially both with the size of domain and with the number of discrete variables that a factor contains. For instance, let the size of the domain be D and the number of child nodes be C, then the number of rows of full CPT is D^C, which is too large for a large domain of discrete variables.Therefore, this approach is not efficient in terms of memory and it cannot work for large hybrid Bayesian networks where factors contain a large number of discrete variables or a huge domain of discrete variables. In addition, the complexity of multiplying is $O(D^{2C})$.

3.1.2. Tree Structured CPT based Operations

The full CPT based multiplying and marginalizing operations may be impossible to carry out for a large scale real-world Bayesian networks due to the finite size of computer memory. Hence, we propose another approach by making use of the decision-tree structure. In this approach, we dynamically reconstruct the decision trees every time after multiplying or marginalizing factors and thus full CPTs are not needed. As an example, let us consider the product of factors using the example from the previous section, Fig. 5. Two trees, named as T_A and T_B, are combined to form a single tree so that all the distinctions that are made both in tree T_A and in tree T_B are made. The scopes of these trees are represented by $\mathbf{C}_A = \text{scope}(T_A)$ and $\mathbf{C}_B = \text{scope}(T_B)$. $\text{Pval}(j, \mathbf{D}, T)$ is used to represent the set of possible combination of values that a set of discrete variables $\mathbf{D}$ can have at the node j in the tree T. First, the leaves of the tree T_A are replaced by the tree T_B as shown in Fig. 6. One can readily confirm that this merged tree can make all the distinction that both trees T_A and T_B make, but it also contains redundant sub-trees that are incompatible with its ancestors in the decision tree. For example, after branching at $D_1 = 1$ and $D_2 = 1$ the right child node specifies $D_2 = 2$ which conflicts with its ancestors. Therefore, we prune the right child node and also remove its corresponding node. This leads to the simplified tree as shown in Fig. 7.

After merging two trees, we can compute the canonical parameters of the new factor. For example, the canonical parameters given $\{D_1, D_2, D_3\} = \{1, 1, 1\}$ are computed as follows:

$$\begin{aligned} K^C &= K^A_{a_1} + K^B_{a_1} \\ \mathbf{h}^C &= \mathbf{h}^A_{a_1} + \mathbf{h}^B_{a_1} \\ g^C &= g^A_{a_1} + g^B_{a_1}. \end{aligned} \tag{20}$$

The algorithm for merging two trees is presented in Table 4.

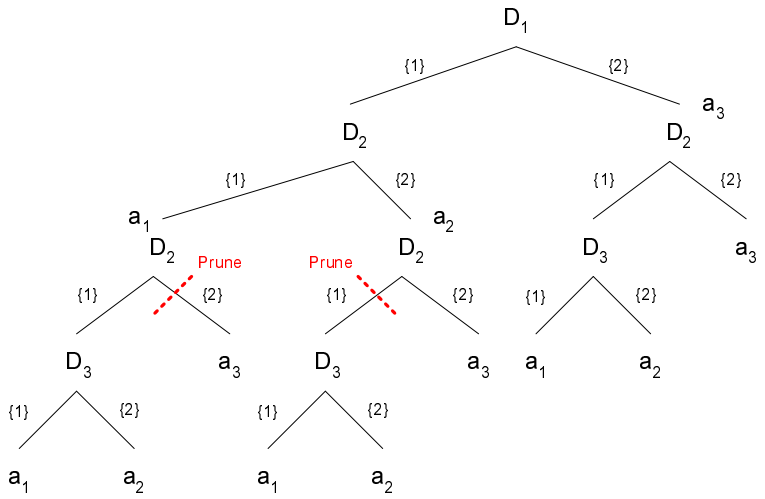


Figure 6: Merging tree T_A and tree T_B

Next, let us consider summing out variables in decision trees. As an example, we consider summing out discrete variables D_1 and D_2 in the decision tree described in Fig. 2. We search for summed out variables D_1 and D_2 from the root node down to the leaf node. In this example, it can be immediately found that the root node is the summed out variable D_1. Therefore, we divide the tree into two sub-trees at the node D_1, then two sub-trees are merged by using $Merge - Trees$ algorithm as shown in Fig. 8.

After that, we continue searching the summed out variables from the root node down to the leaf node in the obtained new tree. We find that the summed out variable D_2 is placed on the root node of the tree and then similar to the previous step, we divide the tree at the node D_2 into two sub-trees and combine them into one tree using *Merge-Trees* algorithm as shown in Fig. 9. Since there is no summed out variable in the current tree, marginalization step of discrete variables is achieved.

With the obtained new tree, weak marginalization is carried out for summing out discrete variables. For instance, the mean vector and covariance matrix of a canonical form $P(D_3 = 1)$ after summing out variables D_1 and

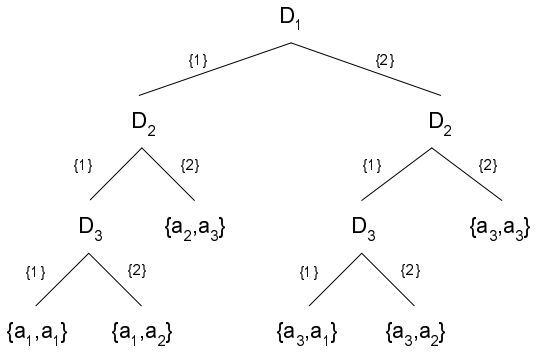


Figure 7: Pruning merged tree

D_2 can be computed as follows:

$$\mu_{\text{new}} = w_1 K_{a_1}^{-1}\mathbf{h}_{a_1} + w_4 K_{a_4}^{-1}\mathbf{h}_{a_4} + 2w_6 K_{a_6}^{-1}\mathbf{h}_{a_6} \tag{21}$$

$$\begin{aligned}\Sigma_{\text{new}} = {} & w_1 K_{a_1}^{-1} + w_4 K_{a_4}^{-1} + 2w_6 K_{a_6}^{-1} + w_1(K_{a_1}^{-1}\mathbf{h}_{a_1} - \mu_{\text{new}})(K_{a_1}^{-1}\mathbf{h}_{a_1}^{C} - \mu_{\text{new}})^T \\ & + w_4(K_{a_4}^{-1}\mathbf{h}_{a_4} - \mu_{\text{new}})(K_{a_4}^{-1}\mathbf{h}_{a_4}^{C} - \mu_{\text{new}})^T \\ & + 2w_6(K_{a_6}^{-1}\mathbf{h}_{a_6} - \mu_{\text{new}})(K_{a_6}^{-1}\mathbf{h}_{a_6}^{C} - \mu_{\text{new}})^T\end{aligned} \tag{22}$$

with

$$\begin{aligned} w_1 &= p(D_3 = 1 | D_1 = \{1,2,3\}, D_2 = \{2,3\}) \\ w_4 &= p(D_3 = 1 | D_1 = \{1,2,3\}, D_2 = \{1,4,5,6\}) \\ w_6 &= p(D_3 = 1 | D_1 = \{4,5,6\}) \end{aligned} \tag{23}$$

.

After summing out the discrete variables, continuous variables are integrated out by using Eq. (19). The detailed algorithm for merging two trees is shown in Table 5.

Above procedure enables us to multiply factors and marginalize over both discrete and continuous variables through dynamically reconstructing decision trees. This method can retain the tree structure since the number of operations and the memory size required to multiply and marginalize factors are much smaller compared to the full CPT based operations. It should be pointed out that the computational cost of marginalization increases exponentially with the number of discrete nodes that we need to sum out. At worst case, the number of summed out nodes are increasing with $O(N^2)$ during marginalization if we cannot prune the redundant nodes. Therefore,

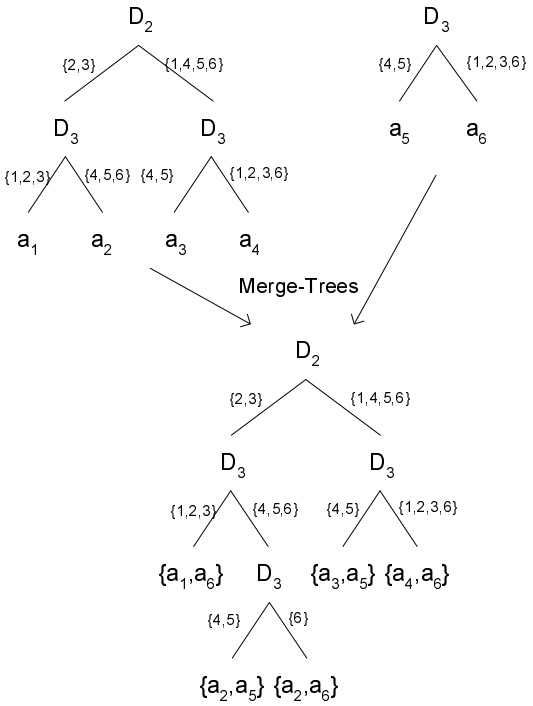


Figure 8: Summing out D_1

depending on the situation, the algorithm may not converge in an acceptable length of time. Nevertheless, compared to the previous approaches, the proposed method allows us to deal with larger domain of discrete variables.

3.1.3. Compact Conditional Probability Table based Operations

Last type of the operations for factor product and marginalization is based on the set of possible combination of values that a set of discrete variables **D** can take at each leaf node. Let us consider again example in Fig. 5 used in the previous section. First we convert the tree structured CPTs into conditional probability tables where each row corresponds to each leaf node and each entry represents a probability distribution as shown in Tables 6 and 7.

This conditional probability table will be called *compact conditional probability table* (compact CPT). Next, we combine two compact CPTs into a single table in order to be able to make all distinctions that original two tables make. Therefore, the compact CPT for factor B is combined with each row of the

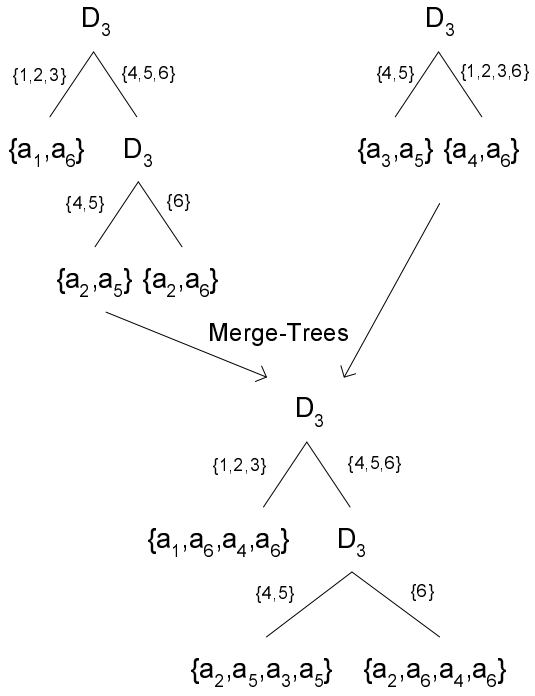


Figure 9: Summing out D_2

compact CPT for factor A as shown in Table 8. After combining two tables, we eliminate inconsistent rows. The canonical table representation of a new factor C of multiplying two factors is computed as shown in Table 9.

Next, let us consider summing out discrete variables D_1 in the compact CPT described in Table 9. We sum up probability distribution to make all distinction except for D_1 that original table makes. For instance, since the first and forth rows of the table are assigned same values of $D_2 = 1$ and $D_3 = 1$, the corresponding two canonical forms are summed up by carrying out weak marginalization that we discussed in the previous section. After summing out discrete variables, we integrate out continuous variables from Eq. (19).

By using compact CPTs, we can compute product and marginalization of factors. Compared to the full CPT based operations, the compact CPTs do not require large memory to carry out these operations. In contrast, the summing out operations need to create groups of rows in the CPTs so that all rows in the each group have the same values for all discrete variables

except for summed out variables and thus the computational cost of grouping grows with $O(N \log N)$. In comparison to the tree structured CPT based operations, the computational cost of the compact CPT based operations does not increase exponentially. One possible drawback is that the compact CPT based operations discard tree structures once they are multiplied or marginalized and thus the decision tree structures are no longer available.

3.2. Belief Propagation Algorithm

In this section, we apply the *belief propagation* (BP) algorithm to handle a large hybrid Bayesian network containing large domain of discrete variables. The four major reasoning patterns in inference are *causal reasoning*, *diagnostic reasoning*, *intercausal reasoning* and *mixed reasoning*. The causal reasoning is to predict the downstream effects of factors from top nodes to bottom nodes while the diagnostic reasoning is to predict 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.

3.2.1. Causal Reasoning

The purpose of causal reasoning is to predict the conditional probability of downstream effects of factors given the evidence of their ancestor variables. Conditional probability can be calculated by a local message passing algorithm known as sum-product algorithm. 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:

$$\tilde{P}_\Phi(\mathbf{C}_i) = \int \sum_{\mathbf{A}_i - A_i} \psi_i \prod_{k \in \mathrm{pa}(i)} P_\Phi(\mathbf{C}_k) dx_1 dx_2 \dots dx_M \tag{24}$$

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 integrated variables and X_i or A_i is the child node of the factor $\mathbf{C}_i$ (If the child node variable is continuous, $A_i = \emptyset$, otherwise $X_i = \emptyset$).

Causal inference is carried out as follows. First, we create a factor for each variable, so the number of factors is equal to the number of nodes in Bayesian networks. A factor corresponding to the top root node variable is a singleton factor that contains the corresponding top root node variable only. Since singleton factors are given evidence due to causal reasoning, cardinalities of their initial and final potentials are set to be 1. Note that a

factor corresponding to the variable that is not placed on the top root node is a non-singleton factor that contains the corresponding variable and its all parent node variables. Therefore, the initial potentials for non-singleton factors are the conditional probability distributions of its corresponding variable given its parent node variables. After creating factors, the sum-product algorithm is carried out to compute the final potentials of all factors. We look for the non-singleton factor for which all parent nodes are assigned final potentials and compute a final potential for the non-singleton factor by using Eq. (24). Since the computed final potential $\tilde{P}_{\Phi}(\mathbf{C}_i)$ is not normalized, the normalized final potential $P_{\Phi}(\mathbf{C}_i)$ is computed every time after computing final potentials. We continue computing the final potentials using the sum-product algorithm until the final potentials of all factors are computed. Because computation of the final potentials is carried out from top node factors to bottom node factors, the probability propagation directions are the same as arcs in Bayesian networks.

3.2.2. Diagnostic and Intercausal Reasoning

Unlike the causal reasoning, conditional probability for diagnostic, intercausal and mixed reasoning cannot be computed by monotonic algorithms. However, the sum-product algorithm can also be utilized for these inferences. Since the computational cost of exact inference algorithm such as VE exponentially increases in the width of the network, the loopy belief propagation (LBP) algorithm is employed in this work. LBP schemes use a cluster graph rather than a clique tree that is used for exact inference (Murphy et al., 1999). Since the constraints defining the clique tree are indispensable for exact inference, the answer of message passing scheme is not correct in LBP.

LBP is carried out as follows: (i) Create the singleton factors for all nodes and intermediate factors for all conditional probability distributions. Therefore, unlike in the causal reasoning, the number of factors is larger than the number of nodes in Bayesian networks. Given the cluster graph, we assign each factor ϕ_k to a cluster $\mathbf{C}_{\alpha(k)}$ so that $\text{scope}(\phi_k) \subseteq \mathbf{C}_{\alpha(k)}$. (ii) Then, the initial potentials ψ_i can be computed as follows:

$$\psi_i = \prod_{k:\alpha(k)=i} \phi_k \tag{25}$$

(iii) Initialize all messages 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:

$$\delta_{i\to j} = \int \sum_{\mathbf{A}_i - \mathbf{S}_{i,j}} \psi_i \prod_{k\in \mathrm{Nb}_i - \{j\}} \delta_{k\to i} dx_1 dx_2 \dots dx_M \tag{26}$$

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 subset of discrete variables, $\mathbf{S'}_{i,j} = \mathbf{X}_i \cap \mathbf{X}_j$ is the subset of continuous variables, and Nb_i is the set of indices of factors that are neighbors of $\mathbf{C}_i$. (iv) The messages are updated by using Eq. (26) in accordance with some message passing schedule until the canonical parameters of all messages are converged. Once the factor receives all messages, final potential $\tilde{P}_\Phi(\mathbf{C}_i)$ can be computed by multiplying them with its initial potential as follows:

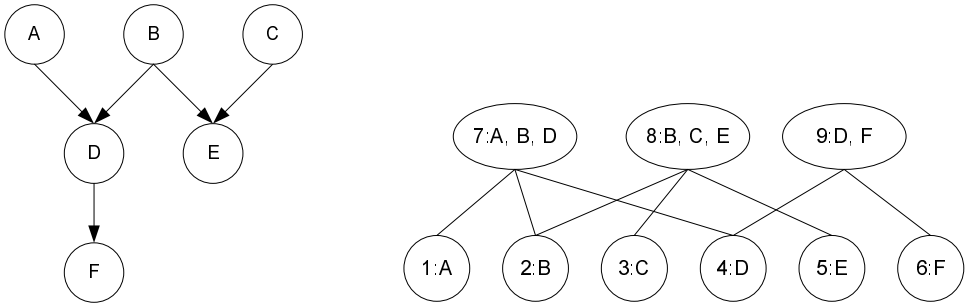
$$\tilde{P}_\Phi(\mathbf{C}_i) = \psi_i \prod_{k\in \mathrm{Nb}_i} \delta_{k\to i}. \tag{27}$$

In order to carry out LBP, we need to create a cluster graph that satisfies the family preservation property. For this purpose, the *Bethe cluster* graph which is a bipartite graph and holds the family preservation property can be used. The first layer of the Bethe cluster graph consists of large clusters $\mathbf{C}_k = \mathrm{scope}(\phi_k)$ while the second layer consists of a singleton cluster X_i for each random variable. Then edges are placed between a large cluster $\mathbf{C}_k$ and a singleton cluster X_i if $X_i \in \mathbf{C}_k$. Let us consider the Bethe cluster graph represented as shown in Fig. 10. This Bethe cluster graph has 6 singleton clusters $\{A\}, \{B\}, \{C\}, \{D\}, \{E\}, \{F\}$ and 3 large clusters $\{A, B, D\}, \{B, C, E\}, \{D, E\}$. Because of $\{B\} \in \{A, B, D\}$ and $\{B\} \in \{B, C, E\}$, we place the edges between cluster 2 and 7 and between cluster 2 and 8.

3.2.3. Belief Propagation with Evidence

Factors need to be modified in accordance with the evidence. Let us consider initializing the factor ψ_i given the discrete evidence $e \in E \cap \Delta$. We delete all canonical parameters $K, \mathbf{h}, g$ in the factor ψ_i that are not consistent with the evidence e.

Next we consider initializing the factor ψ_i given the continuous evidence $e \in E \cap \Gamma$. If the factor does not contain discrete variables, a canonical form of the factor ψ_i is reduced to a context representing the evidence e. In Eq. (17), we set $\mathbf{Y} = y$ with y being the value of the evidence e, then the new



(a) Bayesian Network (b) Bethe Cluster Graph

Figure 10: Example of Bethe Cluster Graph

canonical form given continuous evidence is described as follows (Lauritzen, 1992):

$$\begin{aligned} K' &= K_{\mathbf{XX}} \\ \mathbf{h}' &= \mathbf{h}_{\mathbf{X}} - K_{\mathbf{XY}} y \\ g' &= g + \mathbf{h}_{\mathbf{Y}}^T y - \tfrac{1}{2} y^T \mathbf{K}_{\mathbf{YY}} y. \end{aligned} \tag{28}$$

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

$$g_{a_i} = -\frac{1}{2} y K_{a_i} y + \mathbf{h}_{a_i}^T y + g_{a_i}. \tag{29}$$

Since the new factor contains no continuous variable, the canonical parameters K_{a_i}, h_{a_i} become vacuous. If the factor contains both discrete and continuous variables except for scope(E), the parameters of the canonical form are computed for each instantiation a_i using Eq. (28). After modifying all factors so that all of them are consistent with the evidence, the reasoning algorithms mentioned above can be carried out.

We should note that, even if the compact CPT based operations are employed for factor product and marginalization, the computational task for multiplying incoming messages exponentially increases with the number of incoming messages. This is the case when diagnostic, intercausal or mixed inference is carried out since the LBP is an iterative algorithm which requires a large number of calculation of factor multiplication and marginalization.

In order to reduce the computational cost during LBP, we make use of the property of Bethe cluster graph. In the Bethe cluster graph, the scope of messages is always one variable since there is no edge among any large clusters. If we give the evidence to some variable $X_i = e$, the message departed from the corresponding singleton cluster $\mathbf{C}_i$ never changes during iterations. Therefore, we can fix the messages $\delta_{i \to k}$ as e and thus we can skip the computation described by Eq. (26). Table 10 shows the loopy belief propagation algorithm given evidence.

4. Application Example

4.1. Comparison with SVM and ANN

A simulated example is used to evaluate the validity and performance of the Bayesian networks under the assumption where single model can be used even though we have to deal with any unobservable variables. In this example, we use ANN and SVM for comparison with the presented method. The input and output data are generated from the simple network system where Y_1 is a parent of X_1 and X_2, Y_2 is a parent of X_3 and Z is a parent of Y_1 and Y_2. All variables are linearly dependent when they are connected with the arc. In this example $[X_1, X_2, X_3, Y_1, Y_2]$ are input variables while Z is an output variable, all of which 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 by grid search algorithm. As for ANN, the number of hidden variables is set to be 4, which is determined from cross-validation.

Table 11 shows the computational results. When all variables without output variables can be observed, the accuracies for test data of BN, ANN and SVM are greater than 0.9. Therefore we can say that all methods can accurately predict the output variables. Meanwhile in the case when only input data are partially known, the accuracies of both ANN and SVM are 0.734 and 0.734 respectively. On the other hand, BN predicts the output variable with the high accuracy of 0.848 even when partial input data can be observed. These computational results demonstrate that BN is superior to SVM and ANN in terms of accuracy when input data are partially obtained.

Table 4: Algorithm 1. Merging Trees

Procedure Merge-Trees (
Tree T_A, T_B // Decision trees
)

$\mathbf{S} = \mathbf{C}_A \cap \mathbf{C}_B$ // Sepset of discrete variables among two factors
$T_{\text{new}} = T_A$
Let $a_1, \ldots, a_M$ be indices of leaf nodes in T_{new}
for $i = 1, ..., M$
  $T'_B = T_B$
  if $S \neq \emptyset$
    $List = \emptyset$ // List of next visiting nodes in T_B
    Add the top node of T_B to $List$
    while $List \neq \emptyset$
      if node $List(1)$ has child nodes
        Let l and r be the left and right child nodes of the node $List(1)$
        if $\text{Pval}(a_i, \mathbf{S}, T_A) \cap \text{Pval}(\text{List}(l), \mathbf{S}, T_B) = \emptyset$
          and $\text{Pval}(a_i, \mathbf{S}, T_A) \cap \text{Pval}(\text{List}(r), \mathbf{S}, T_B) \neq \emptyset$
          Replace current node $List(1)$ with node r and remove node l
          Add node r to the end of $List$
        elseif $\text{Pval}(a_i, \mathbf{S}, T_A) \cap \text{Pval}(\text{List}(l), \mathbf{S}, T_B) \neq \emptyset$
          and $\text{Pval}(a_i, \mathbf{S}, T_A) \cap \text{Pval}(\text{List}(r), \mathbf{S}, T_B) = \emptyset$
          Replace current node $List(1)$ with node l and remove node r
          Add node l to the end of $List$
        else
          Add l and r to the end of $List$
      Delete $List(1)$
  Replace the nodes a_i of tree T_{new} with the new structure T'_B
return T_{new}

Table 5: Algorithm 2. Summing out Tree

Procedure Sum-Out-Tree (
Tree T // Decision tree
A set of discrete variables $\mathbf{V}$ // Summed out variables
)
1 $T_{\text{new}} = T$
2 **while** scope$(T_{\text{new}}) \cap \mathbf{V} \neq \emptyset$
3 Let i be the top node of scope$(T_{\text{new}}) \cap \mathbf{V}$
4 Make sub-trees T_A and T_B whose root nodes are the child nodes of node i
5 T_{new} = Merge-Trees (T_A, T_B)
6 **return** T_{new}

Table 6: Compact CPT for Factor A

D_1	D_2	Probability Distribution
1	1	$C(\mathbf{X}; K^A_{a_1}, \mathbf{h}^A_{a_1}, g^A_{a_1})$
1	2	$C(\mathbf{X}; K^A_{a_2}, \mathbf{h}^A_{a_3}, g^A_{a_2})$
2	1,2	$C(\mathbf{X}; K^A_{a_3}, \mathbf{h}^A_{a_2}, g^A_{a_3})$

Table 7: Compact CPT for Factor B

D_2	D_3	Probability Distribution
1	1	$C(\mathbf{X}; K^B_{a_1}, \mathbf{h}^B_{a_1}, g^B_{a_1})$
1	2	$C(\mathbf{X}; K^B_{a_2}, \mathbf{h}^B_{a_2}, g^B_{a_2})$
2	1,2	$C(\mathbf{X}; K^B_{a_3}, \mathbf{h}^B_{a_3}, g^B_{a_3})$

Table 8: Combined Compact CPT of Factor A and B

Factor A		Factor B		Probability
D_1	D_2	D_2	D_3	Distribution
1	1	1	1	$C(\mathbf{X}; K^C_{a_1}, \mathbf{h}^C_{a_1}, g^C_{a_1}) = C(\mathbf{X}; K^A_{a_1}, \mathbf{h}^A_{a_1}, g^A_{a_1}) \cdot C(\mathbf{X}; K^B_{a_1}, \mathbf{h}^B_{a_1}, g^B_{a_1})$
		1	2	$C(\mathbf{X}; K^C_{a_2}, \mathbf{h}^C_{a_2}, g^C_{a_2}) = C(\mathbf{X}; K^A_{a_1}, \mathbf{h}^A_{a_1}, g^A_{a_1}) \cdot C(\mathbf{X}; K^B_{a_2}, \mathbf{h}^B_{a_2}, g^B_{a_2})$
		2	1,2	inconsistent
1	2	1	1	inconsistent
		1	2	inconsistent
		2	1,2	$C(\mathbf{X}; K^C_{a_3}, \mathbf{h}^C_{a_3}, g^C_{a_3}) = C(\mathbf{X}; K^A_{a_2}, \mathbf{h}^A_{a_2}, g^A_{a_2}) \cdot C(\mathbf{X}; K^B_{a_3}, \mathbf{h}^B_{a_3}, g^B_{a_3})$
2	1,2	1	1	$C(\mathbf{X}; K^C_{a_4}, \mathbf{h}^C_{a_4}, g^C_{a_4}) = C(\mathbf{X}; K^A_{a_3}, \mathbf{h}^A_{a_3}, g^A_{a_3}) \cdot C(\mathbf{X}; K^B_{a_1}, \mathbf{h}^B_{a_1}, g^B_{a_1})$
		1	2	$C(\mathbf{X}; K^C_{a_5}, \mathbf{h}^C_{a_5}, g^C_{a_5}) = C(\mathbf{X}; K^A_{a_3}, \mathbf{h}^A_{a_3}, g^A_{a_3}) \cdot C(\mathbf{X}; K^B_{a_2}, \mathbf{h}^B_{a_2}, g^B_{a_2})$
		2	1,2	$C(\mathbf{X}; K^C_{a_6}, \mathbf{h}^C_{a_6}, g^C_{a_6}) = C(\mathbf{X}; K^A_{a_3}, \mathbf{h}^A_{a_3}, g^A_{a_3}) \cdot C(\mathbf{X}; K^B_{a_2}, \mathbf{h}^B_{a_2}, g^B_{a_2})$

Table 9: Compact CPT for Factor C

D_1	D_2	D_3	Probability Distribution
1	1	1	$C(\mathbf{X}; K^C_{a_1}, \mathbf{h}^C_{a_1}, g^C_{a_1}) = C(\mathbf{X}; K^A_{a_1}, \mathbf{h}^A_{a_1}, g^A_{a_1}) \cdot C(\mathbf{X}; K^B_{a_1}, \mathbf{h}^B_{a_1}, g^B_{a_1})$
1	1	2	$C(\mathbf{X}; K^C_{a_2}, \mathbf{h}^C_{a_2}, g^C_{a_2}) = C(\mathbf{X}; K^A_{a_1}, \mathbf{h}^A_{a_1}, g^A_{a_1}) \cdot C(\mathbf{X}; K^B_{a_2}, \mathbf{h}^B_{a_2}, g^B_{a_2})$
1	2	1,2	$C(\mathbf{X}; K^C_{a_3}, \mathbf{h}^C_{a_3}, g^C_{a_3}) = C(\mathbf{X}; K^A_{a_2}, \mathbf{h}^A_{a_2}, g^A_{a_2}) \cdot C(\mathbf{X}; K^B_{a_3}, \mathbf{h}^B_{a_3}, g^B_{a_3})$
2	1	1	$C(\mathbf{X}; K^C_{a_4}, \mathbf{h}^C_{a_4}, g^C_{a_4}) = C(\mathbf{X}; K^A_{a_3}, \mathbf{h}^A_{a_3}, g^A_{a_3}) \cdot C(\mathbf{X}; K^B_{a_1}, \mathbf{h}^B_{a_1}, g^B_{a_1})$
2	1	2	$C(\mathbf{X}; K^C_{a_5}, \mathbf{h}^C_{a_5}, g^C_{a_5}) = C(\mathbf{X}; K^A_{a_3}, \mathbf{h}^A_{a_3}, g^A_{a_3}) \cdot C(\mathbf{X}; K^B_{a_2}, \mathbf{h}^B_{a_2}, g^B_{a_2})$
2	2	1,2	$C(\mathbf{X}; K^C_{a_6}, \mathbf{h}^C_{a_6}, g^C_{a_6}) = C(\mathbf{X}; K^A_{a_3}, \mathbf{h}^A_{a_3}, g^A_{a_3}) \cdot C(\mathbf{X}; K^B_{a_2}, \mathbf{h}^B_{a_2}, g^B_{a_2})$

Table 10: Algorithm 3. Hybrid loopy belief propagation algorithm

Procedure Hybrid-LBP (
Φ // Set of factors
e // Evidence
$Cgraph$ // Bethe cluster graph
)
1 Set $\mathcal{E}$ to be set of edges in $Cgraph$
2 Initialize all factors so that all of them are consistent with the evidence **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** true
5 Select $(i, j) \in \mathcal{E}$
6 **if** scope(ψ_i) = scope(e)
7 **continue**
8 Update message $\mu_{i \to j}$ from Eq. (26)
9 **if** all messages are converged
10 **break**
11 Compute final potential $\tilde{P}_\Phi(\mathbf{C}_i)$ from (27)
12 **return** $\tilde{P}_\Phi(\mathbf{C}_i)$

Table 11: Prediction accuracies 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

4.2. Steel Production Process

In this study, the real-world steel production data are utilized to examine the performance of the proposed hybrid inference method. All 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). Steel production is managed via two major tasks, production planning and production scheduling. At the production planning stage, the sales department receives orders from the customers and considers customer demand, plant profit, production capacity, inventory and shipment dates. For this purpose, the production planning task requires a model that can predict production loads and processing times required to meet the customer demands. Once orders are finalized, the manufacturing department makes a production schedule that satisfies customer deadlines and production capacity constraints. Similar to the production planning task, this task also requires a model that can predict the production load and process time. At the production scheduling stage, additional information about operating conditions is also known.

Due to complexity of the manufacturing operations, potential for product defects which need to be rectified, and also the reality that exact production times for new grades of steel plates (which have not been produced before and need to be produced to meet a new customer order), the exact production times are not known. We apply the methodology described in the previous sections to construct the most likely structure of the Bayesian network representing the steel plates manufacturing process. This network is then used to estimate most likely production time for each grade of steel plates. Once the production times are known (estimated), one can proceed to plan and schedule the production.

In this application example, we create the Bayesian network that represents the relationship among system variables including customer demands, operating conditions, production loads and a process time. With the constructed Bayesian network, production loads and a process time are estimated

so that they are consistent with the evidence such as customer demands or operating conditions.

To construct the Bayesian network, 21 discrete variables and 5 continuous variables are selected as shown in Table 12.

Table 12: 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(discritized)
7	Customer demand G	10(discritized)
8	Customer demand H	10(discritized)
9	Customer demand I	10(discritized)
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

The variables related to the customer demands include continuous variables such as size of plates and strength and they are converted into discrete variables because each node of these variables has a discrete child node and the canonical tables cannot represent factors containing continuous nodes with a discrete child node. The variables related to the operating condition are all discrete variables such as heat and primer conditions. The variable corresponding to each production load is an integer variable; if the corresponding production process is not needed, it is 1, otherwise it is 2.

4.3. Inference Results

First, we design the Bayesian network structure that represents relationship among system variables from historical process data shown in Fig. 11. The method to learn network structure from historical dataset is described in Appendix A.

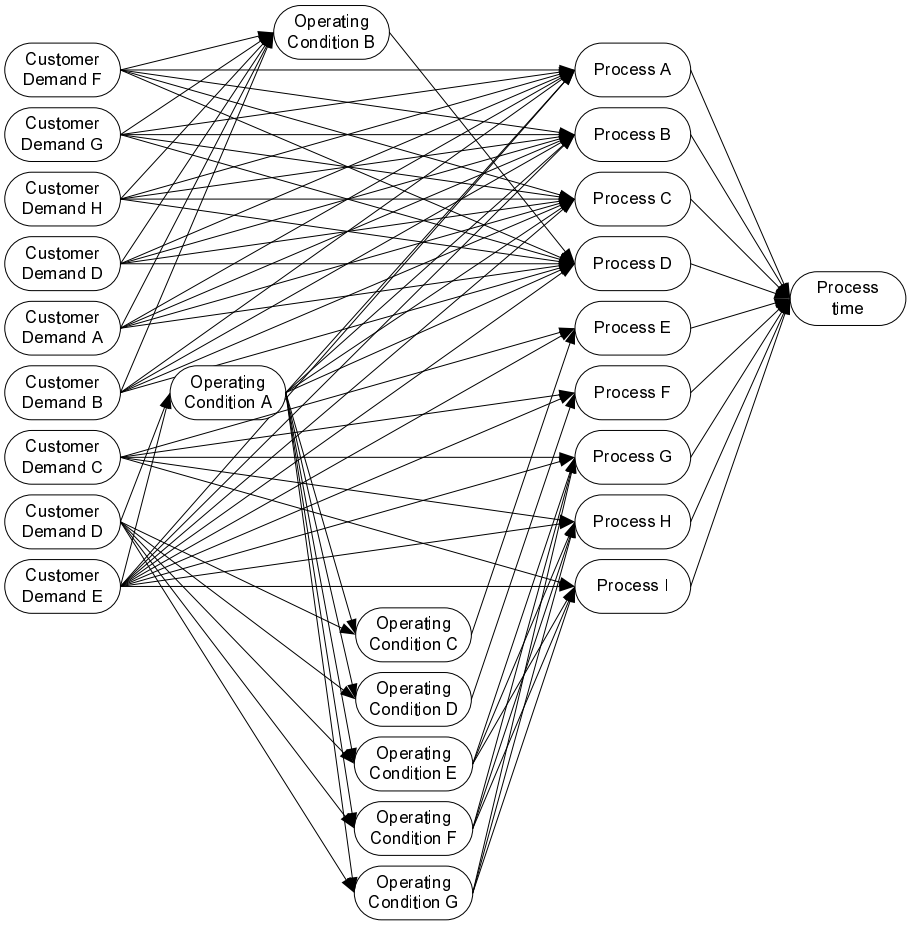


Figure 11: Bayesian network for steel production process

With the designed network structure, the decision-tree structured CPTs are computed from historical process data. The traditional simple CPT method cannot be used in this application example for the following reason. Let us consider the non-singleton factor that contains variables of customer

demand A, B, E, F, G, H, operating condition A, and process A. Since the product of cardinalities of all variables is $\times 12 \times 2 \times 20 \times 10 \times 10 \times 10 \times 10 \times 5 \times 2 = 48$ million, it would require a huge amount of training data to compute the appropriate parameters of all elements in the table. Among the proposed three types of multiplying and marginalizing operations, we employ the compact CPTs based operations. That is because the full CPT based operations would require a huge amount of memory and the tree-structured CPT based operations needs a large execution time for marginalization due to a size of decision tree. It should be pointed out that since we use the compact CPTs, the decision-tree structure of final potentials cannot be obtained.

4.3.1. Case 1: Variables of both customer demands and operating conditions are known and the variables of both production loads and process time are unknown

In this case, we need to estimate the production loads and process time given all evidence except for them. Such situation typically arises at the production scheduling stage. This types of inference is causal reasoning. The training data set consisting of $100,000$ steel plates is used to learn the decision-tree structured CPTs. The test data set of other $100,000$ plates is used to evaluate prediction accuracy. We compute the true probability distributions for comparison on each production group, which contains plates that have same customer demands and operating conditions. We select the production groups that have more than 200 plates for this purpose, while the remaining production groups are discarded because these production groups do not have a sufficient number of plates to compute the actual probability distributions. Figs. 12 and 13 show the prediction results of the production loads. It can be seen that there is little difference between predicted and actual probability distributions in all production loads. Compared to Case 2, the prediction accuracy of production loads for process D, process E, process F, process G, process H and process I is significantly better. This is mainly due to the fact that we know the values of the operating conditions which have a significant influence on these production loads. Fig. 14 shows the means and variances of the inferred and actual process time and KL divergences between them. One can see that KL divergences are smaller than the second test case. For further improvement, it may be desirable to add other variables such as the amount of in-process inventory, the number of workers and the failure rate of each machine, which will have an impact on the process time. However, these kinds of discussion are outside the scope of our research. The

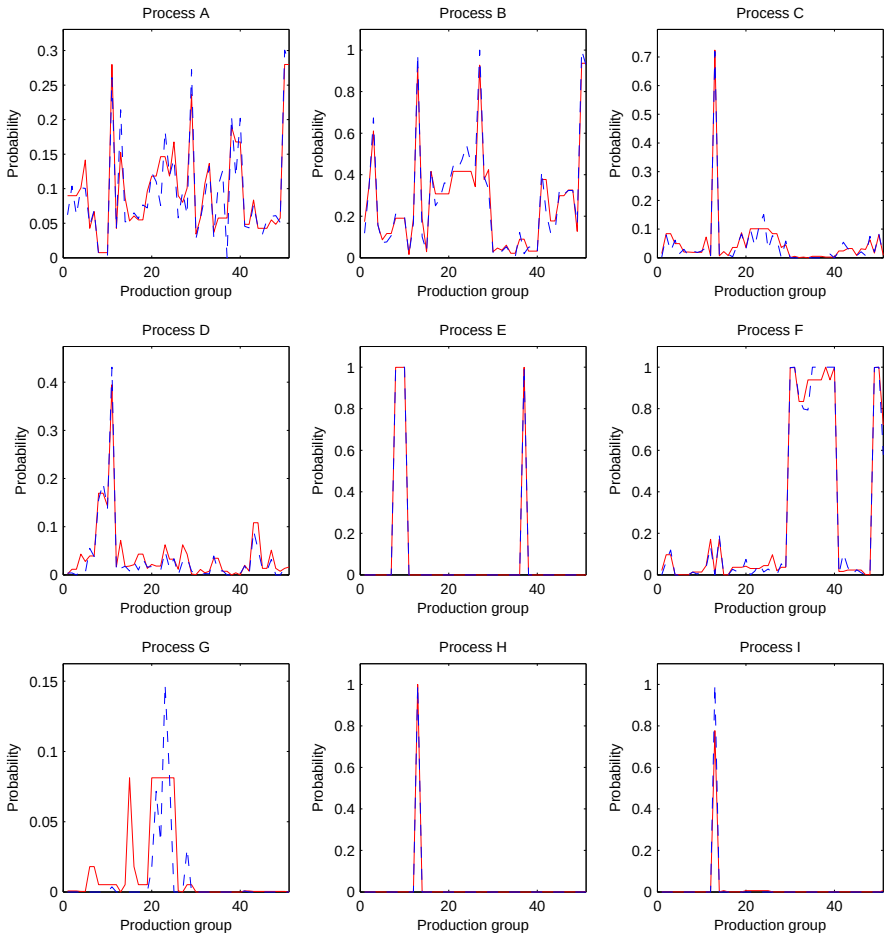


Figure 12: Probability of the predicted production loads given customer demands and operating conditions in case 1

average execution time for the inference in all production group is $9.60[s]$, which is almost same as the second test scenario. This example demonstrates that, even though our Bayesian network includes a large domain of discrete variables, the causal reasoning can be carried out by using the proposed decision-tree structured conditional probability table representations.

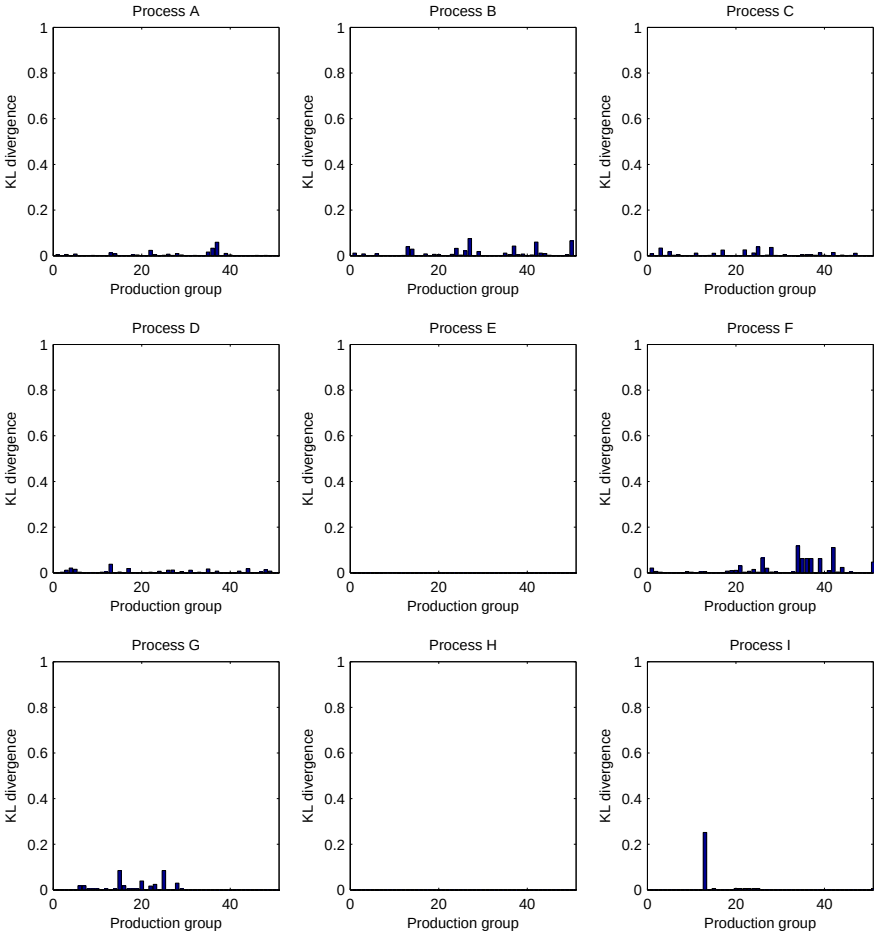


Figure 13: KL divergence between the predicted and true probability distributions of production loads given customer demands and operating conditions in case 1

4.3.2. Case 2: Variables representing the customer demands are known and the other variables are unknown

This situation arises typically when persons in the sales department receive orders from customers and want to know the production loads and the process times for these orders. Since the purpose of this inference is to pre-

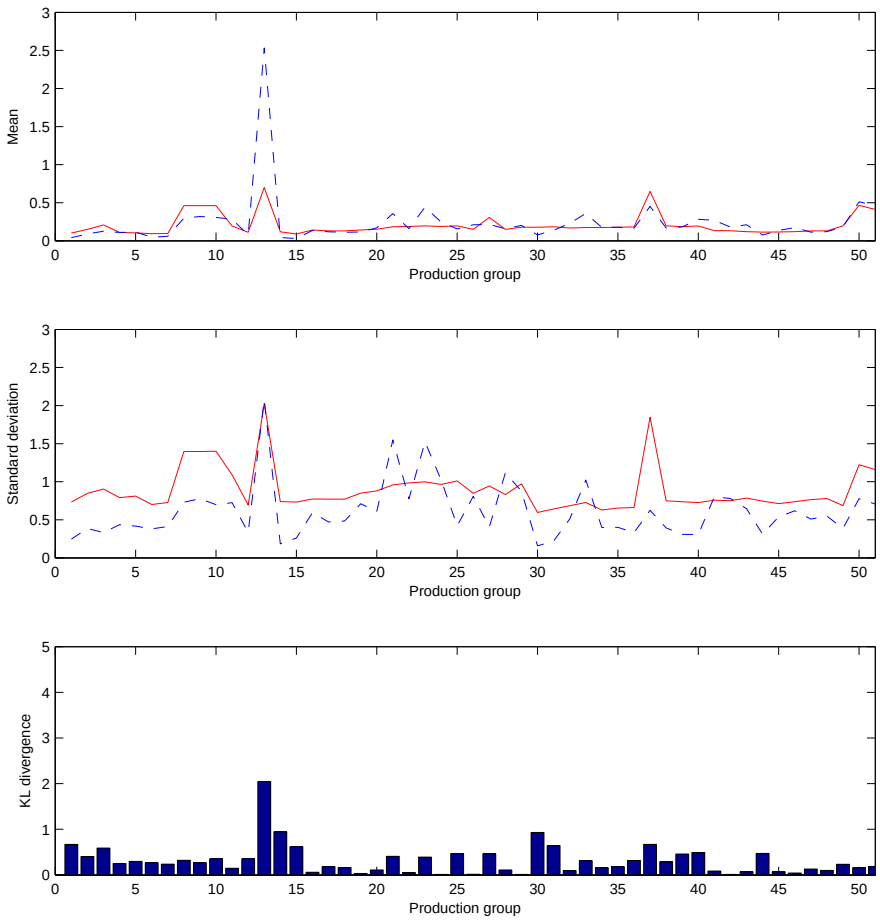


Figure 14: Mean, standard deviation and KL divergence between the predicted and true probability distributions of process time given customer demands and operating conditions in case 1

dict the effects of production loads and the process time given the causes of the customer demands, this type of inference is causal reasoning. Compared to Case 1, there are more uncertainties here, since the production loads are not known and one can expect reduced certainty of predictions. We use the same decision-tree structured CPTs that was learned in Case 1. The

probabilities of the values of production loads being 2 and their KL divergences between the predicted and actual probability distributions are shown in Figs. 15 and 16. The red solid lines represent the predicted values while

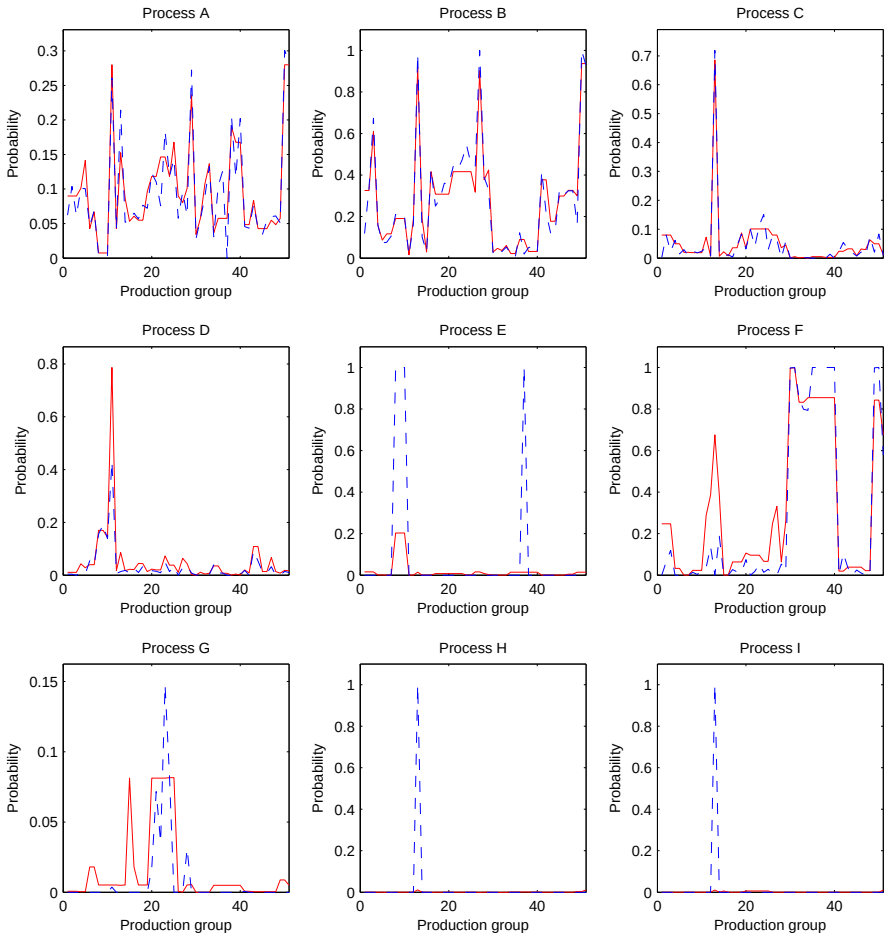


Figure 15: Probability of the predicted production loads given customer demands in case 2

the blue dash lines denote the actual values. One can readily observe that the probability distributions of process A, process B, process C and process

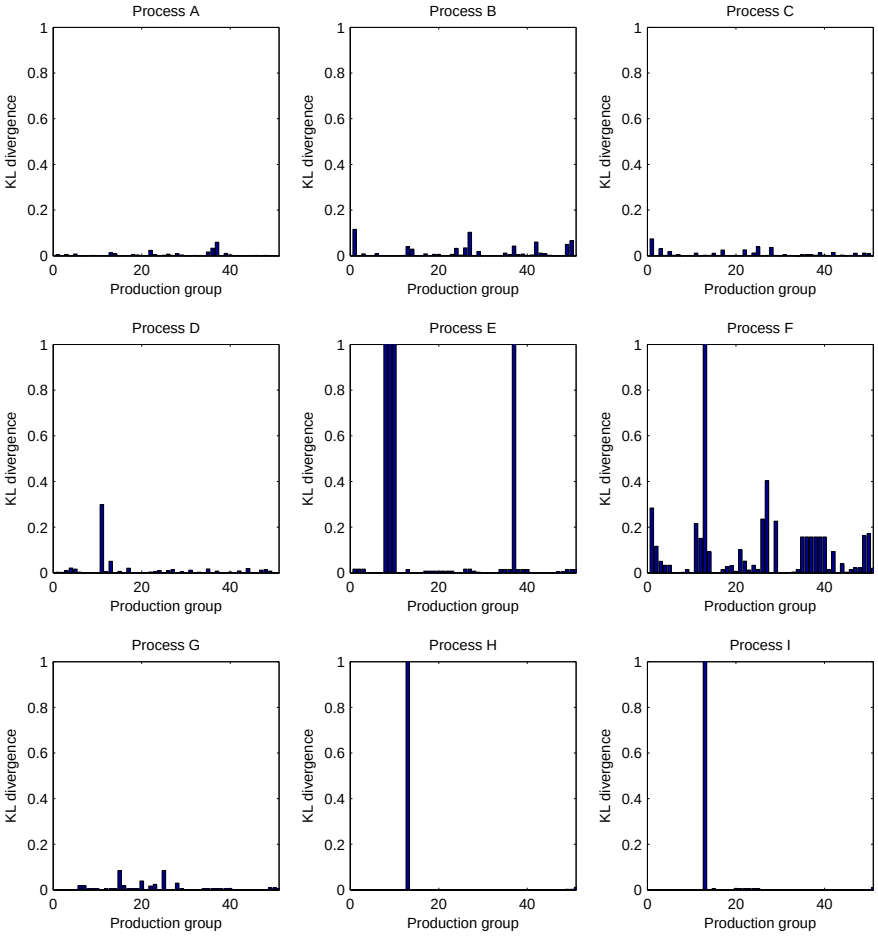


Figure 16: KL divergence between the predicted and true probability distributions of production loads given customer demands in case 2

D are very close to actual ones. However, the results of the other production loads such as process E, process F, process G, process H and process I show the bad prediction performance. This is most likely due to the fact that these production loads are significantly affected by the operating conditions, which are assumed to be unknown in this test case. In addition, the predic-

tion results for the process time are shown in Fig. 17. It can be seen that

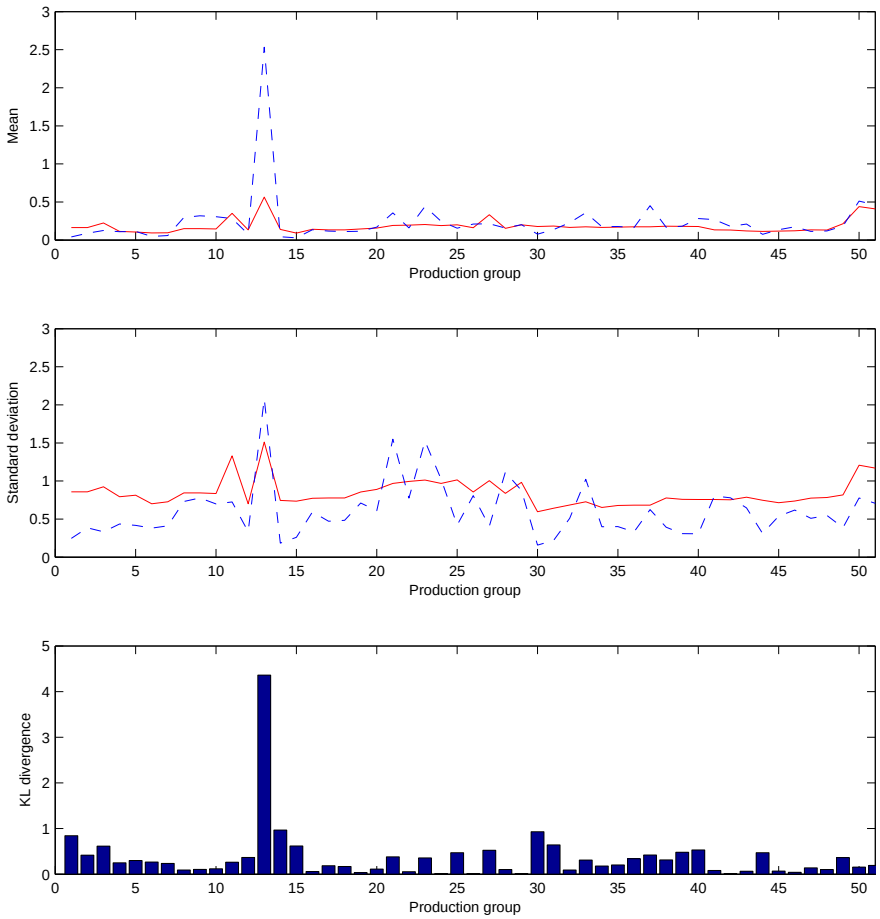


Figure 17: Mean, standard deviation and KL divergence between the predicted and true probability distributions of process time given customer demands in case 2

the inferred probability distributions are largely different from the true ones. That is because the process time is computed from the final potentials of production loads and the probability distributions of some production loads are not accurately predicted. The average execution time for the inference

in all production group is 9.96[s].

Results of test cases 1 and 2 indicate that we can carry out the most plausible reasoning from the values of known variables using a single Bayesian network model. This is very useful particularly in instances when available variables are different at different steps. Having a single Bayesian network model avoids the need to have multiple specific models tailored for specific purposes, which causes model maintenance issues and lack of model consistency.

4.3.3. Case 3: Customer demands and all production loads are known while the operating conditions are unknown

This kind of is required when we want to decide which operating conditions to use in production so that both the customer demands and production capacity can be satisfied. Since this type of the inference is mixed reasoning, the proposed hybrid LBP algorithm is employed. Similarly to the previous two cases, we evaluate the prediction accuracy for each production that contains steel plates that have the same customer demands and production loads. We have selected the production groups containing more than 100 plates to compute the true probability distributions. The computed probability distributions of production loads are shown in Fig. 18. Different colors of the lines are used to represent different values of each variable while the solid lines denote the predicted probabilities and the dashed lines represent the actual probabilities. In addition, the KL divergences between the predicted and actual probability distributions are shown in Fig. 19. It can be seen that most KL divergences are very small and the proposed inference algorithm can accurately predict the probability distributions of unknown variables. Therefore, we can say that the proposed inference method can carry out the mixed reasoning even though this Bayesian network has a large domain of discrete variables. The average execution time for the inference in all production groups is 1.91[s].

5. Conclusion

Complex multi-stage manufacturing processes, such as steel plates manufacturing, have many manufacturing steps which are not fully deterministic. Production planning and scheduling for such systems requires that probabilities of specific outcomes at each production step be accounted for. Business decisions, e.g. committing to deliver a specific order, scheduling production,

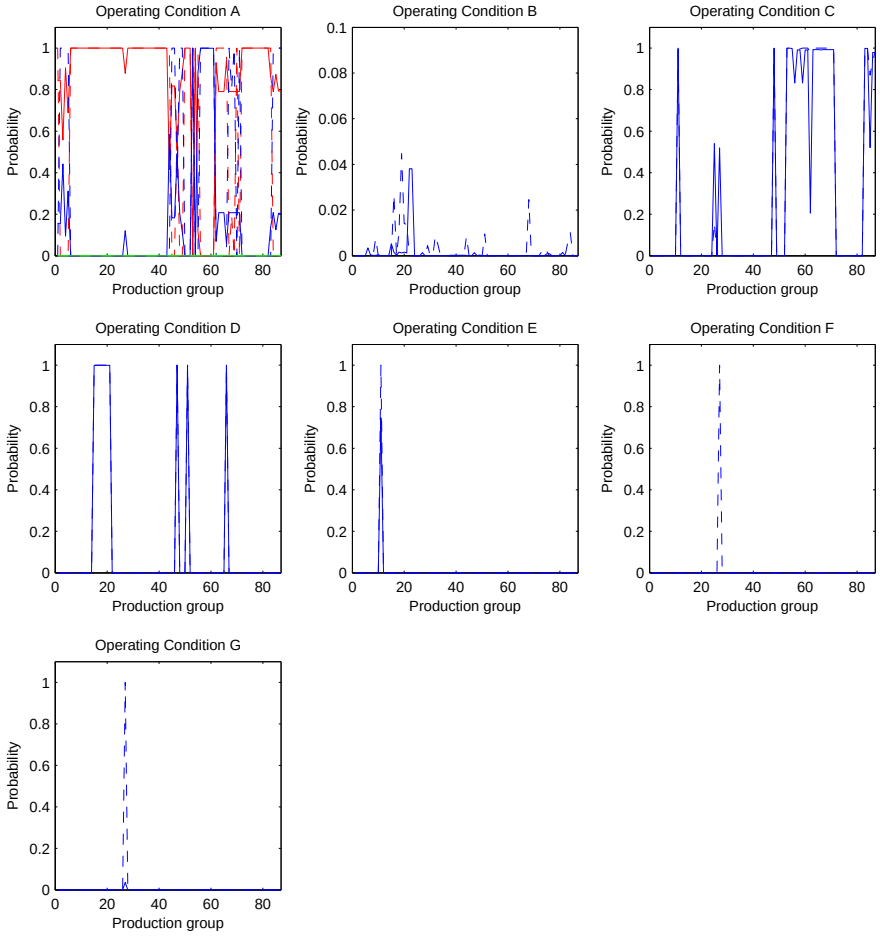


Figure 18: Probability of operating conditions in case 3

or deciding on operating conditions, require different types of reasoning to conclude the most likely outcomes. Instead of using different models for different types of reasoning, we have used the same inference technique for all decision making. The method employs decision-tree structured conditional probability tables (CPTs) based hybrid inference technique for the hybrid Bayesian network containing large domain discrete variables. The main con-

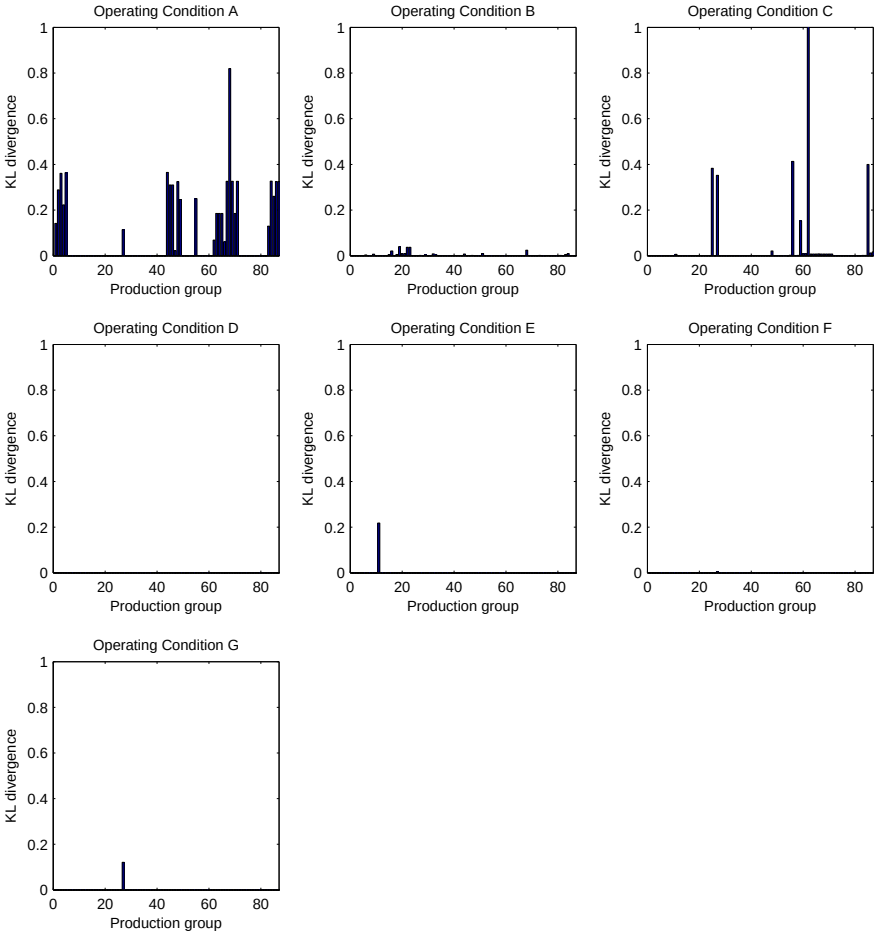


Figure 19: KL divergence of operating conditions in case 3

tribution of this research is new methodology to construct the context-specific CPTs represented by decision trees through classification tree algorithm. We have introduced three new types of operators for computing multiplication and marginalization of factors represented by decision-tree structure based CPTs. These proposed operations enable us to carry out inference by using belief propagation algorithms. For causal reasoning, the direction of proba-

bility propagation is downstream from the top node variables to the bottom nodes variables. Since other types of reasoning cannot be carried out using such monotonic logic based algorithm, we use more complicated algorithm, i.e. the loopy belief propagation. In order to reduce the computational cost, we fix the values of messages that depart from the nodes given evidence and skip the computation of these messages during iterations.

In conclusion, our method is able to handle the large domain discrete variables without increasing computational cost exponentially. In addition, in our approach, we can discretize the continuous parents as finely as needed, which does not increase the number of parameters in intermediate factors due to decision-tree structured CPTs. Meanwhile, a possible weakness is that the proposed method assumes that continuous variables are Gaussian due to canonical form, while approximate inference algorithm such as Monte Carlo method can handle any kinds of distribution. The other weakness may be that If we have extremely large domain size of discrete child (more than 100 domain cardinalities), the factors also become large, which may make the tree-CPTs sparse because our approach cannot split the discrete child nodes.

Real life steel production process data have been used to examine the effectiveness of the proposed inference algorithm. The results demonstrate that the proposed algorithm accurately and very rapidly predicts the probability distributions of unobserved variables in large hybrid Bayesian networks.

Future work will focus on applying the presented Bayesian network model to the scheduling and planning for steelmaking plants such that both the profit and customer satisfaction are maximized.

Appendix A. Structure Learning

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 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. In

this paper, we employ BIC scores described as follows:

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

where $\text{pa}_{\mathcal{G}}(x_i)$ are the parent nodes of x_i given graph $\mathcal{G}$, $\mathbf{I}(x_i;\text{pa}_{\mathcal{G}}(x_i))$ 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. With this score function, the optimal graph $\mathcal{G}^*$ can be computed as follows:

$$\mathcal{G}^* = \text{argmax}_{\mathcal{G}}\text{score}_{\text{BIC}}(\mathcal{G}:\mathcal{D}). \quad \text{(A.2)}$$

Since this combinational optimization problem is known to be *NP*-hard (**?**), it is difficult to compute 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. One can easily noticed that the nodes of variables belonging to the customer demands should be placed on the root nodes in the Bayesian network while production loads are affected by the customer demands, operating conditions or both and not vice versa. Taking into account above considerations, we propose the type of Bayesian networks as shown in Fig. A.20.

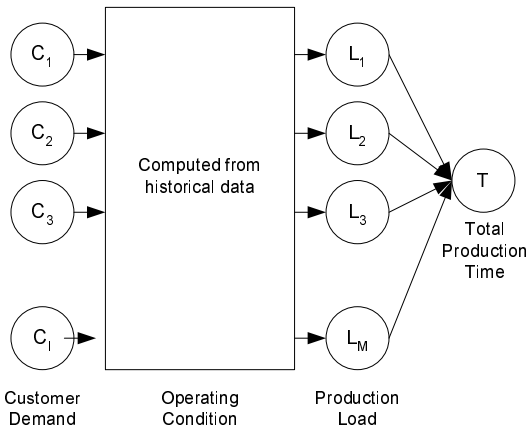


Figure A.20: Bayesian network

References

Bishop, C., 2006. Pattern Recognition and Machine Learning. Springer-Verlag, New York, NY.

Boutilier, C., Friedman, N., Goldszmidt, M., Koller, D., 1996. Context-specific independence in Bayesian networks. In: Proceedings of UAI-96. pp. 115–123.

Breiman, L., Friedman, J., Olshen, R., Stone, C., 1984. Classication and Regression Trees. Wadsworth International Group, Belmont.

Buyukozkan, G., Kayakutlu, G., Karakadlar, I., 2015. Assessment of lean manufacturing effect on business performance using bayesian belief networks. Expert Systems with Applications 42, 6539–6551.

Cai, Z., Sun, S., Si, S., Yannou, B., 2011. Identifying product failure rate based on a conditional bayesian network classifier. Expert Systems with Applications 38 (5036-5043).

Cano, A., Gómez-Olmedo, M., Moral, S., Pérez-Ariza, C. B., 2014. Extended probability trees for probabilistic graphical models. probabilistic graphical models. Probabilistic Graphical Models, 113–128.

Chavira, M., Darwiche, A., 2007. Compiling bayesian networks using variable elimination. In: IJCAI'07 Proceedings of the 20th international joint conference on Artifical intelligence. pp. 2443–2449.

Chickering, D., Heckerman, D., Meek, C., 1997. A bayesian approach to learning Bayesian networks with local structure. In: Proceedings of UAI-97. pp. 80–89.

Cobb, B., Rumi, R., Salmeron, A., 2007. Bayesian Network Models with Discrete and Continuous Variables. SpringerVerlag, Berlin.

DesJardins, M., Rathod, P., 2008. Learning structured Bayesian networks: Combining abstraction hierarchies and tree-structured conditional probability tables. Comput. Intell. 24, 1–22.

Friedman, N., Goldszmidt, M., 1996. Learning Bayesian networks with local structure. In: Proceedings of UAI-96. pp. 252–262.

Gogate, V., Domingos, P., 2013. Structured message passing. In: Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence (UAI2013). pp. 252–261.

Koller, D., Friedman, N., 2009. Probabilistic Graphical Models: Principles and Techniques. MIT Press.

Koller, D., Lerner, U., Angelov, D., 1999. A general algorithm for approximate inference and its application to hybrid bayes nets. In: Proceedings of UAI-99. pp. 324–333.

Kozlov, A., Koller, D., 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., Jensen, F., 2001. Stable local computation with conditional gaussian distribution. Stat. Comput. 11, 191–203.

Lauritzen, S., Wermuth, N., 1989. Graphical models for associations between variables, some of which are qualitative and some quantitative. Ann. Stat. 17, 31–57.

Melo, A. C. V., Sanchez, A. J., 2008. Software maintenance project delays prediction using bayesian networks. Expert Systems with Applications 34, 908–919.

Moral, S., Rumi, R., Salmern, A., 2001. Mixtures of Truncated Exponentials in Hybrid Bayesian Networks. SpringerVerlag, Berlin.

Murphy, K., Y.Weiss, Jordan, M., 1999. Nonuniform dynamic discretization in hybrid networks. In: Proceedings of UAI-99. pp. 467–475.

Pearl, J., 1988. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan-Kaufmann, San Francisco.

Perkusich, M., Soares, G., 2014. A procedure to detect problems of processes in software development projects using bayesian networks. Expert Systems with Applications 42, 437–450.

Poole, D., Zhang, N., 2003. Exploiting contextual independence in probabilistic inference. J. Artif. Intell. Res. 18, 263–313.

Rumi, R., Salmeron, A., 2007. Approximate probability propagation with mixtures of truncated exponentials. Int. J. Approx. Reason. 45, 191–210.

Sharma, R., Poole, D., 2003. Efficient inference in large discrete domains. In: Proceedings of UAI-2003. pp. 535–542.

Yuan, C., Druzdzel, M. J., 2006. Hybrid loopy belief propagation. In: Proceedings of the Third European Workshop on Probabilistic Graphical Models (PGM'06). pp. 317–324.