The Marginal Factorization of Bayesian Networks and Its Application

Dan Wu,1,* Michael Wong2,†

1School of Computer Science, University of Windsor, Windsor, Ontario N9B 3P4, Canada
2Department of Computer Science, University of Regina, Regina, Saskatchewan S4S 0A2, Canada

A Bayesian network consists of a directed acyclic graph (DAG) and a set of conditional probability distributions (CPDs); they together define a joint probability distribution (jpd). The structure of the DAG dictates how a jpd can be factorized as a product of CPDs. This CPD factorization view of Bayesian networks has been well recognized and studied in the uncertainty community. In this article, we take a different perspective by studying a marginal factorization view of Bayesian networks. In particular, we propose an algebraic characterization of equivalent DAGs based on the marginal factorization of a jpd defined by a Bayesian network. Moreover, we show a simple method to identify all the compelled edges in a DAG. © 2004 Wiley Periodicals, Inc.

1. INTRODUCTION

The Bayesian network model has been well established as a tool for managing uncertainty using probability. A standard Bayesian network consists of two components: (i) a graphical structure, namely, a directed acyclic graph (DAG), and (ii) a set of conditional probability distributions (CPDs), each of which is associated with a node in the DAG, such that the product of these CPDs defines a joint probability distribution (jpd). In other words, a Bayesian network (BN) can be regarded as a graphical representation of a jpd being factorized as a product of CPDs.

As a definitive component of the Bayesian network model, the CPDs and the jpd factorization (in terms of CPDs) have been extensively used and studied. However, the study of the factorization of a jpd in terms of marginals has received considerably less attention, except perhaps the decomposable Markov network model. In this article, we take a different perspective to study Bayesian networks by focusing on the marginal factorization of a jpd. Our study reveals a nontrivial

*Author to whom all correspondence should be addressed: e-mail: danwu@uwindsor.ca.
†e-mail: wong@cs.uregina.ca.

result that cannot be easily seen by studying either the graphical DAG structure or the CPD factorization. More specifically, we demonstrate and prove that the marginal factorization of a jpd defined by a Bayesian network is in fact an algebraic characterization of an equivalence class of DAGs. Moreover, based on this algebraic characterization, we show a simple algebraic method for identifying all the compelled edges in a DAG.

The article is organized as follows. The background knowledge is given in Section 2. In Section 3, we propose an algebraic characterization of equivalent DAGs based on the marginal factorization of a jpd defined by a Bayesian network. We demonstrate an application of the algebraic characterization proposed by showing a simple method for identifying compelled edges in Section 4. We conclude the article in Section 5.

2. BACKGROUND

Let $U = \{a_1, a_2, \ldots, a_n\}$ denote a finite set of discrete variables. We usually use a lowercase letter to represent a single variable and an uppercase letter to represent a set of variables. A single lowercase letter can also be regarded as a singleton set when appropriate and no confusion arises. We use $p(U)$ to denote a joint probability distribution (jpd) over $U$. We call $p(V)$, where $V \subseteq U$, a marginal (distribution) of $p(U)$ and $p(X|Y)$ a conditional probability distribution (CPD). The CPD $p(X|Y)$ is defined as:

$$p(X|Y) = \frac{p(XY)}{p(Y)},$$

where $p(Y) > 0$, $X \subseteq U$, and $Y \subseteq U$. By $XY$, we mean $X \cup Y$ (similarly, by $ab$, we mean $\{a\} \cup \{b\}$). In Equation 1, we say that the denominator $p(Y)$ is “absorbed” by the numerator $p(XY)$ to yield the CPD $p(X|Y)$. In general, if $W \subseteq V$, then $p(W)$ can be absorbed by $p(V)$ to yield $p(V - W|W)$.

**Definition 1.** A Bayesian network (BN) over a set $U = \{a_1, a_2, \ldots, a_n\}$ of variables consists of two components:

(i) A directed acyclic graph (DAG) $\mathcal{D}$. There is a one-to-one correspondence between the nodes in $\mathcal{D}$ and the variables in $U$. Each node in $\mathcal{D}$ represents a variable $a_i \in U$. The parents of a node $a_i$ in $\mathcal{D}$ are denoted as $pa(a_i)$.

(ii) A set $\{p(a_i|pa(a_i))| i = 1, \ldots, n\}$ of CPDs. Each node $a_i$ in $\mathcal{D}$ is associated with a CPD $p(a_i|pa(a_i))$.

These two components define a jpd $p(U)$ as follows:

$$p(U) = \prod_{i=1}^{n} p(a_i|pa(a_i)).$$
We refer to the product in Equation 2 as the Bayesian factorization of $p(U)$ (with respect to $D$), and we further call the jpd $p(U)$ a Bayesian jpd.

Because a BN is always associated with its corresponding DAG, we thus will use the terms BN and DAG interchangeably if no confusion arises.

It is important to note that given the Bayesian factorization in Equation 2, the DAG of a BN can be drawn by directing an arrow from each variable in $pa(a_i)$ to $a_i$.

Example 1. Consider the “Asia” BN defined over the set $U$ of variables where $U = \{a, b, c, d, e, f, g, h\}$. Its DAG $D$ is shown in Figure 1, and the corresponding set of CPDs is $C = \{p(a), p(b), p(c|a), p(d|b), p(e|b), p(f|cd), p(h|f), p(g|ef)\}$. These two components define the jpd $p(U)$ of this Bayesian network, namely:

$$p(U) = p(a) \cdot p(b) \cdot p(c|a) \cdot p(d|b) \cdot p(e|b) \cdot p(f|cd) \cdot p(h|f) \cdot p(g|ef).$$ (3)

The CPD factorization in Equation 3 is a Bayesian factorization of $p(U)$.

3. THE MARGINAL FACTORIZATION OF BAYESIAN NETWORKS—AN ALGEBRAIC CHARACTERIZATION OF EQUIVALENT DAGs

A Bayesian factorization can be considered as a jpd $p(U)$ being factorized as a product of CPDs. In other words, a Bayesian factorization is in fact a CPD factorization. The CPDs and the Bayesian factorization of a jpd $p(U)$ play important roles in the research of Bayesian networks and have received much attention. For instance, the CPDs in a Bayesian factorization participate in the local computation for probabilistic inference. The Bayesian factorization is used in variable elimination for probabilistic inference. In learning Bayesian networks, besides learning the graphical DAG structure of a BN, another important task is to learn the numerical parameters for each CPD designated by the DAG.

On the other hand, by definition, a CPD $p(X|Y)$ can be expressed in terms of marginals as $p(XY)/p(Y)$; therefore, one can always transform a Bayesian factorization into a marginal factorization.
Example 2. Consider the Bayesian factorization in Equation 3; we can easily transform it into a marginal factorization as follows:

\[
p(U) = p(a) \cdot p(b) \cdot p(c|a) \cdot p(d|b) \cdot p(e|b) \cdot p(f|cd) \cdot p(h|f) \cdot p(g|ef),
\]

\[
= p(a) \cdot p(b) \cdot \frac{p(ca) \cdot p(db) \cdot p(eb) \cdot p(fcd) \cdot p(hf) \cdot p(gef)}{p(a) \cdot p(b) \cdot p(b) \cdot p(cd) \cdot p(f) \cdot p(ef)},
\]

\[
= \frac{p(ca) \cdot p(db) \cdot p(eb) \cdot p(fcd) \cdot p(hf) \cdot p(gef)}{p(b) \cdot p(cd) \cdot p(f) \cdot p(ef)}.
\]

Perhaps because obtaining a marginal factorization from a Bayesian factorization is trivial, the study on the marginal factorization of a BN has not been pursued. In this section, we will demonstrate that the marginal factorization of a BN is strongly related to the research on equivalent DAGs and it can actually serve as an algebraic characterization of an equivalence class of DAGs. We begin with a short review of equivalence classes of DAGs.

### 3.1. Equivalent Classes of DAGs and Their Graphical Characterizations

It has been noted that different DAGs may define the same Bayesian jpd and thus encode the same conditional independence information. This observation motivated the study of equivalence classes of DAGs.

**Definition 2.** Two DAGs \( \mathcal{D} \) and \( \mathcal{D}' \) defined over the same set \( U \) of variables are equivalent if for every jpd \( p(U) \) that can be factorized as a Bayesian factorization with respect to \( \mathcal{D} \), \( p(U) \) can also be factorized as a Bayesian factorization with respect to \( \mathcal{D}' \), and vice versa.

We use \( \mathcal{D}_1 \cong \mathcal{D}_2 \) to denote that two DAGs \( \mathcal{D}_1 \) and \( \mathcal{D}_2 \) are equivalent; “\( \cong \)” is in fact an equivalence relation. We can therefore group DAGs defined over a set \( U \) of variables into different equivalence classes.

Many graphical criteria can be used to determine if two DAGs are equivalent. Verma and Pearl graphically characterized equivalent DAGs using the notions of skeleton and \( v \)-structure.

**Definition 3.** A \( v \) - structure in a DAG \( \mathcal{D} \) is an ordered triple of nodes \( (x, y, z) \) such that (1) \( \mathcal{D} \) contains the edges \( \langle x, y \rangle \) and \( \langle z, y \rangle \), and (2) \( x \) and \( z \) are not directly connected in \( \mathcal{D} \).

**Definition 4.** The skeleton of a DAG is the undirected graph obtained by dropping the directionality of every edge in the DAG.

**Theorem 1.** Two DAGs are equivalent if and only if they have the same skeleton and the same \( v \)-structures.

\(^a\)We use the notation \( \langle x, y \rangle \) to denote a directed edge from \( x \) to \( y \) in a DAG.
Example 3. Consider the four DAGs shown in Figure 2a–d. It can be verified that they all have the same skeleton as shown in Figure 2e. Moreover, all the DAGs have the same v-structure, which is $(b, e, c)$. Therefore, these four DAGs are equivalent to each other.

Chickering\(^5\) suggested characterizing equivalent DAGs using the notion of covered edges.

**Definition 5.** An edge $\langle x, y \rangle$ is covered in a DAG if $pa(y) = pa(x) \cup \{x\}$.

Example 4. Consider the edge $\langle x, y \rangle$ in the DAG in Figure 3; it can be easily verified that $pa(x) = \{z_1, z_2\}$ and $pa(y) = \{x, z_1, z_2\}$. Therefore, $pa(y) = pa(x) \cup \{x\}$ and $\langle x, y \rangle$ is a covered edge.

**Theorem 2.** Let $\mathcal{D}$ be a DAG containing an edge $\langle x, y \rangle$ and $\mathcal{D}'$ be a DAG identical to $\mathcal{D}$ except that the edge $\langle x, y \rangle$ in $\mathcal{D}$ is oriented as $\langle y, x \rangle$ in $\mathcal{D}'$; $\mathcal{D} \approx \mathcal{D}'$ if and only if $\langle x, y \rangle$ is a covered edge in $\mathcal{D}$.

Because of the symmetry relationship between $\mathcal{D}$ and $\mathcal{D}'$, we immediately have the following corollary.

Figure 2. Equivalent DAGs and their skeletons.

Figure 3. The edge $\langle x, y \rangle$ is a covered edge.
Corollary 1. \( \langle y, x \rangle \) is also a covered edge in \( D' \).

Chickering further showed the following property of covered edges.

**Theorem 3** \(^5\) A property holds for every pair of equivalent DAGs that differ by a single covered edge orientation if and only if that property holds for all the DAGs in the equivalence class.

In other words, Theorem 3 says that a given property is invariant over all equivalent DAGs if and only if we can show that this property is invariant to every pair of DAGs that differ by a single reversal of covered edge.

### 3.2. An Algebraic Characterization of Equivalence Classes of DAGs

In the following, we propose a novel algebraic characterization of an equivalence class of DAGs based on the marginal factorization of the corresponding Bayesian jpd. We begin with an illustrative example.

**Example 5.** Consider the DAGs shown in Figure 2. We use \( D_a, D_b, D_c, \) and \( D_d \) to refer to the equivalent DAGs in Figure 2a,b,c,d, respectively. We can write down the Bayesian factorization for each of the DAGs as follows:

\[
D_a: p(abcdef) = p(a) \cdot p(b|a) \cdot p(c|a) \cdot p(d|b) \cdot p(e|bc),
\]
\[
D_b: p(abcdef) = p(b) \cdot p(a|b) \cdot p(c|a) \cdot p(d|b) \cdot p(e|bc),
\]
\[
D_c: p(abcdef) = p(c) \cdot p(a|c) \cdot p(b|a) \cdot p(d|b) \cdot p(e|bc),
\]
\[
D_d: p(abcdef) = p(d) \cdot p(b|d) \cdot p(a|b) \cdot p(c|a) \cdot p(e|bc).
\]

Obviously, although all four DAGs are equivalent, their Bayesian factorizations are different. However, if we transform the above Bayesian factorizations into marginal factorizations, we will obtain the following results:

\[
D_a: p(abcde) = p(a) \cdot \frac{p(ba)}{p(a)} \cdot \frac{p(ca)}{p(a)} \cdot \frac{p(db)}{p(b)} \cdot \frac{p(ebc)}{p(bc)} = \frac{p(ba) \cdot p(ca) \cdot p(db) \cdot p(ebc)}{p(a) \cdot p(b) \cdot p(bc)}, \tag{4}
\]
\[
D_b: p(abcde) = p(b) \cdot \frac{p(ab)}{p(b)} \cdot \frac{p(ca)}{p(a)} \cdot \frac{p(db)}{p(b)} \cdot \frac{p(ebc)}{p(bc)} = \frac{p(ba) \cdot p(ca) \cdot p(db) \cdot p(ebc)}{p(a) \cdot p(b) \cdot p(bc)}, \tag{5}
\]
It can be easily seen that the final marginal factorizations in Equations 4, 5, 6, and 7 are identical.

The above example demonstrates that equivalent DAGs, although graphically different, converge algebraically to the same marginal factorization of the Bayesian jpd. In the following, we generalize this observation and propose an algebraic characterization of equivalence classes of DAGs.

**Definition 6.** Consider a BN defined over a set \( U = \{x_1, \ldots, x_n\} \) of variables with its Bayesian factorization and marginal factorization as follows in Equations 8 and 10, respectively:

\[
p(U) = p(x_1) \cdot p(x_2|pa(x_2)) \cdot \ldots \cdot p(x_n|pa(x_n)),
\]

\[
= \frac{p(x_1)}{1} \cdot \frac{p(x_2, pa(x_2))}{p(pa(x_2))} \cdot \ldots \cdot \frac{p(x_n, pa(x_n))}{p(pa(x_n))},
\]

\[
= \prod_{i,j} \frac{p(x_i, pa(x_i))}{p(pa(x_j))},
\]

where \( \{x_i, pa(x_i)\} \neq \{pa(x_j)\} \) for any \( 1 \leq i, j \leq n \). The marginal factorization of \( p(U) \) obtained in Equation 10 by canceling every applicable numerator and denominator in Equation 9 is called the intrinsic factorization of the BN.

For instance, the marginal factorization in Equation 4 is the intrinsic factorization of the BN whose DAG is in Figure 2a. It is worth noting that the number of denominators in the intrinsic factorization is always less than the number of numerators.

Before we present the algebraic characterization of equivalence classes of DAGs, we need to establish the following lemma first.

**Lemma 1.** If \( D \) is a DAG containing a covered edge \( (x, y) \) and \( D' \) is a DAG identical to \( D \) except that the edge \( (x, y) \) in \( D \) is now oriented as \( (y, x) \) in \( D' \), then (a) \( pa'(y) = pa(x) \), and (b) \( \{x\} \cup pa'(x) = \{y\} \cup pa(y) \), where \( pa'(y) \) is the parent set for the node \( y \) in \( D' \), \( pa(x) \) is the parent set for the node \( x \) in \( D \).
Because \((x, y)\) is a covered edge in \(D\), it follows \(pa(y) = \{x\} \cup pa(x)\). Similarly, by Corollary 1, \((y, x)\) is a covered edge in \(D'\); it then follows \(pa'(x) = \{y\} \cup pa'(y)\). Because \((x, y)\) is the only edge that differs in \(D\) and \(D'\), it then follows \(pa'(x) = pa(x) \cup \{y\}, pa'(y) = pa(y) - \{x\}\). It then immediately follows that \(pa'(y) = pa(y) - \{x\} = (\{x\} \cup pa(x)) - \{x\} = pa(x), \) and \(\{x\} \cup pa'(x) = \{x\} \cup \{y\} \cup pa'(y) = \{x\} \cup \{y\} \cup pa(x) = \{y\} \cup (\{x\} \cup pa(x)) = \{y\} \cup pa(y)\).

We are now ready to present the algebraic characterization that characterizes and describes equivalence classes of DAGs.

**Theorem 4.** Two DAGs are equivalent if and only if they define the same intrinsic factorization.

**Proof.** The proof for the “if” part is trivial. If two BNs have the same intrinsic factorization, they must define the same jpd. We thus only need to prove the “only if” part.

According to Theorem 2, we only need to show our claim holds for two equivalent DAGs, say \(D_1\) and \(D_2\), that differ by a single covered edge orientation. We further assume that in \(D_1\), it has a covered edge \((x, y)\), whereas in \(D_2\), it has the edge \((y, x)\), which is also a covered edge by Corollary 1. Because \(D_1\) and \(D_2\) are equivalent, by Definition 2, this means that for every jpd \(p(U)\) that can be factorized with respect to \(D_1\), it can also be factorized with respect to \(D_2\), and vice versa. Consider such a jpd \(p(U)\) and its Bayesian factorizations with respect to both \(D_1\) and \(D_2\); because they differ only in the edge between \(x\) and \(y\), for every other node \(z\), where \(z \neq x \neq y\), in both the Bayesian factorizations of \(D_1\) and \(D_2\), we should have the CPD \(p(z|pa(z))\). Therefore, we can write down the Bayesian factorization for \(D_1\) as follows:

\[
p(U) = p(x|pa(x)) \cdot p(y|pa(y)) \cdot \prod_{z \neq x, z \neq y} p(z|pa(z)). \tag{11}
\]

Similarly, we can write down the Bayesian factorization for \(D_2\) as follows:

\[
p(U) = p(x|pa'(x)) \cdot p(y|pa'(y)) \cdot \prod_{z \neq x, z \neq y} p(z|pa(z)). \tag{12}
\]

If we turn the Bayesian factorization in Equation 11 into a marginal factorization, we obtain

\[
p(U) = \frac{p(x, pa(x))}{p(pa(x))} \cdot \frac{p(y, pa(y))}{p(pa(y))} \cdot \prod_{z \neq x, z \neq y} \frac{p(z, pa(z))}{p(pa(z))}. \tag{13}
\]

Similarly, if we turn the Bayesian factorization in Equation 12 into a marginal factorization, we obtain

\[
p(U) = \frac{p(x, pa'(x))}{p(pa'(x))} \cdot \frac{p(y, pa'(y))}{p(pa'(y))} \cdot \prod_{z \neq x, z \neq y} \frac{p(z, pa(z))}{p(pa(z))}. \tag{14}
\]
As one may see, the only difference between the two marginal factorizations in Equations 13 and 14 are the CPDs for nodes \( x \) and \( y \). By the definition of covered edge, because \( (x, y) \) is a covered edge in \( D_1 \), \( pa(y) = \{x\} \cup pa(x) \), it follows that \( p(pa(y)) = p(x, pa(x)) \), and we can simplify the marginal factorization in Equation 13 as follows:

\[
p(U) = \frac{p(y, pa(y))}{p(pa(x))} \cdot \prod_{z \neq x, z \neq y} \frac{p(z, pa(z))}{p(pa(z))}.
\] (15)

By the same reason that \( (y, x) \) is a covered edge in \( D_2 \), \( pa'(x) = \{y\} \cup pa'(y) \), it follows that \( p(pa'(y)) = p(x, pa'(x)) \), and we can simplify the marginal factorization in Equation 14 as follows:

\[
p(U) = \frac{p(x, pa'(x))}{p(pa'(y))} \cdot \prod_{z \neq x, z \neq y} \frac{p(z, pa(z))}{p(pa(z))}.
\] (16)

By Lemma 1, \( pa'(y) = pa(x) \) and \( \{x\} \cup pa'(x) = \{y\} \cup pa(y) \); this implies that \( p(pa'(y)) = p(pa(x)) \) and \( p(x, pa'(x)) = p(y, pa(y)) \). It then follows immediately that the marginal factorizations for \( D_1 \) and \( D_2 \) as shown in Equations 15 and 16, respectively, are identical. Therefore, \( D_1 \) and \( D_2 \) have the same intrinsic factorization.

The following corollary follows immediately.

**Corollary 2.** All the DAGs in the same equivalence class have the same intrinsic factorization.

Theorem 4 and Corollary 2 indicate that the intrinsic factorization for an equivalence class of DAGs is unique and it characterizes and describes an equivalence class of DAGs algebraically.

Note that the intrinsic factorization is in fact the marginal factorization of a BN. This study of algebraic characterization of equivalence classes of DAGs clearly demonstrates the usefulness of the marginal factorization of a BN and it gives an algebraic insight of equivalence classes of DAGs other than those graphical characterizations.

### 4. AN APPLICATION OF THE ALGEBRAIC CHARACTERIZATION OF EQUIVALENCE CLASSES OF DAGs

In this section, we describe and suggest a method for identifying compelled edges in a DAG based on its intrinsic factorization. Although different DAGs in the same equivalence class have different graphical structures, certain directed edges retain their directionality in all those equivalent DAGs. These edges are referred to as compelled edges. It is useful to identify these compelled edges in learning a BN from observed data, and Chickering proposed a graphical algorithm for identifying such edges.\(^5\)
Definition 7. An edge \((x, y)\) in a DAG \(\mathcal{D}\) is a compelled edge if, for each \(\mathcal{D}' = \mathcal{D}\), \((x, y)\) is also in \(\mathcal{D}'\). Otherwise, it is a reversible edge.

Example 6. It can be verified that the DAGs shown in Figure 4 comprise an equivalence class by Theorem 4 and Corollary 2 as the intrinsic factorizations for each of the DAGs are identical. Furthermore, the edges \((b, e)\), \((c, e)\), \((e, g)\), \((e, h)\), \((f, h)\), \((e, i)\), and \((f, i)\) appear in all of the equivalent DAGs; therefore, they are compelled edges. On the other hand, the edge \((a, c)\), for example, is reversible.

4.1. The Motivation

The idea behind our proposed method for identifying compelled edges in a DAG is very intuitive. As will be demonstrated shortly, the intrinsic factorization of a given DAG algebraically represents the whole equivalence class and it contains all the information needed to restore each equivalent DAG in the class (more precisely, to restore its Bayesian factorization from which the equivalent DAG can be drawn). It is this restoration process that motivates the development of the proposed method for identifying compelled edges. We begin with an illustrative example.

Example 7. Suppose we want to find out all the compelled edges in the DAG shown in Figure 4a. We first obtain its intrinsic factorization as follows:

\[
p(U) = \frac{p(b) \cdot p(ca) \cdot p(da) \cdot p(efc) \cdot p(ge) \cdot p(hef) \cdot p(ief)}{p(a) \cdot p(bc) \cdot p(c) \cdot p(e) \cdot p(ef) \cdot p(ef)}.
\]

We now demonstrate how we can restore each equivalent DAG (actually its Bayesian factorization) in the equivalence class characterized by the above intrinsic factorization.
To transform the intrinsic factorization in Equation 17 into a Bayesian factorization, we need to absorb all the denominators in Equation 17 as follows:

$$p(U) = \frac{p(b)}{p(a)} \cdot \frac{p(ca)}{1} \cdot \frac{p(da)}{p(bc)} \cdot \frac{p(ebc)}{p(c)} \cdot \frac{p(fc)}{p(e)} \cdot \frac{p(ge)}{p(ef)} \cdot \frac{p(ge)}{p(e)} \cdot \frac{p(hef)}{p(ef)} \cdot \frac{p(ief)}{p(e)}$$  

$$= p(b) \cdot p(c|a) \cdot p(da) \cdot p(e|bc) \cdot p(f|c) \cdot p(g|e) \cdot p(h|ef) \cdot p(i|ef)$$  

$$= p(b) \cdot p(c|a) \cdot p(d|a) \cdot p(e|bc) \cdot p(f|c) \cdot p(g|e) \cdot p(h|ef) \cdot p(i|ef)$$  

or

$$p(U) = \frac{p(b)}{p(c)} \cdot \frac{p(ca)}{p(a)} \cdot \frac{p(da)}{p(bc)} \cdot \frac{p(ebc)}{p(c)} \cdot \frac{p(fc)}{p(e)} \cdot \frac{p(ge)}{p(ef)} \cdot \frac{p(ge)}{p(e)} \cdot \frac{p(hef)}{p(ef)} \cdot \frac{p(ief)}{p(e)}$$  

$$= p(b) \cdot p(c) \cdot p(d|a) \cdot p(e|bc) \cdot p(f|c) \cdot p(g|e) \cdot p(h|ef) \cdot p(i|ef)$$  

or

$$p(U) = \frac{p(b)}{1} \cdot \frac{p(ca)}{p(c)} \cdot \frac{p(da)}{p(a)} \cdot \frac{p(ebc)}{p(bc)} \cdot \frac{p(fc)}{p(e)} \cdot \frac{p(ge)}{p(ef)} \cdot \frac{p(ge)}{p(e)} \cdot \frac{p(hef)}{p(ef)} \cdot \frac{p(ief)}{p(e)}$$  

$$= p(b) \cdot p(a|c) \cdot p(d|a) \cdot p(e|bc) \cdot p(f|c) \cdot p(g|e) \cdot p(h|ef) \cdot p(i|ef).$$

In Equations 18, 21, and 23, we have absorbed each denominator in the intrinsic factorization by an appropriate numerator; these absorptions resulted in Equations 20, 22, and 24, respectively. We thus have finally obtained three different DAGs, which exactly correspond to the three equivalent DAGs shown in Figure 4b,c,d, respectively. Therefore, we have successfully restored all the DAGs that are equivalent to the one given in Figure 4a.

There are a few important remarks regarding the above demonstration in Example 7.

**Remark 1.** Different DAGs in the same equivalence class are obtained in the example, depending on how each denominator in the intrinsic factorization is absorbed. It is obvious that different absorption will result in different Bayeisan factorizations, hence, produce different but equivalent DAGs.

**Remark 2.** It is noted that during the course of absorbing denominators, some denominator is “forced” to be absorbed by a particular fixed numerator, no other choices. For instance, the denominator $p(bc)$ has to be absorbed by the numerator $p(ebc)$, no other choices.
Before we move on, let us scrutinize the denominator absorption we have made in the example. Recall that for a denominator \( p(X) \) to be absorbed by a numerator \( p(Y) \), it must be the case that \( X \subseteq Y \). Under this constraint, the possible absorption of each denominator can be summarized by the following expressions:

\[
\begin{align*}
    a & \Rightarrow \{ca, da\} \\
    bc & \Rightarrow \{efc\} \\
    c & \Rightarrow \{ca, ebc, fc\} \\
    e & \Rightarrow \{ef, ge, hef, ief\} \\
    ef & \Rightarrow \{hef, ief\}
\end{align*}
\]

where \( X = \{Y_1, \ldots, Y_n\} \) means that a denominator \( p(X) \) can possibly be absorbed by \( p(Y_1), \ldots, p(Y_n) \). We further call the set \( \{Y_1, \ldots, Y_n\} \), denoted \( AS(X) \), the absorption set for \( X \). Note that \( Y_i \in AS(X) \) if \( X \subseteq Y_i \), and we say that \( Y_i \) is a candidate numerator to absorb \( X \).

We will use \( |AS(X)| \) to denote the cardinality of \( AS(X) \). It is noted that the set

\[
D = \{X_i \mid p(X_i) \text{ is a denominator in the intrinsic factorization}\}
\]

is a multiset, and we will use \( |X_i| \) to denote the number of occurrences of \( X_i \) in \( D \).

Recall that the DAG of a BN can be drawn based on its Bayesian factorization. More precisely, for each CPD \( p(a_i \mid pa(a_i)) \), we can draw a directed edge from each node in \( pa(a_i) \) to node \( a_i \). Following this line of reasoning, if a denominator, say \( p(X) \), has no choice but is “forced” to be absorbed by a fixed numerator, say \( p(Y) \), to yield the CPD \( p(Y - X \mid X) \), then \( p(Y - X \mid X) \) will appear in every possible resulting Bayesian factorization after absorbing all the denominators in the intrinsic factorization. Therefore, it is expected that the edges \( (x, y) \), where \( y \in Y - X, x \in X \), will appear in every resulting equivalent DAG, and they are compelled edges by definition. We will use the notation \( E_X^Y = \{ (x, y) \mid y \in Y - X, x \in X \} \) to represent all such edges drawn. For instance, in expression 26, the only numerator that is applicable to absorb \( p(\text{bc}) \) is \( p(\text{e}\text{bc}) \); hence, the CPD \( p(e \mid \text{bc}) \) will be obtained and appear in every possible resulting Bayesian factorization. Hence, the edges in \( E_{\text{bc}}^e \), namely, \( \langle b, e \rangle \) and \( \langle c, e \rangle \), will appear in all possible resulting equivalent DAGs, which implies that they are compelled edges as can be verified by Figure 4. Special attention should be paid to the two identical denominators \( p(\text{ef}) \) and \( p(\text{ef}) \) in expressions 29 and 30, respectively. Although these two denominators are syntactically identical, both of them have to be absorbed to obtain a Bayesian factorization. The applicable numerators for both of them are the absorption set \( \{hef, ief\} \), which contains exactly two elements—the same number of the number of the occurrence of \( p(\text{ef}) \) as denominators. This indicates that one \( p(ef) \) must be absorbed by \( p(hef) \) and the other \( p(ef) \) must be absorbed by \( p(ief) \), no other choices. These absorptions imply that the CPDs \( p(h \mid ef) \) and \( p(i \mid ef) \) will be obtained and appear in every possible resulting Bayesian factorization, and the edges in \( E_{\text{ef}}^{\text{hef}} \) and \( E_{\text{ef}}^{\text{ief}} \), namely, \( \langle e, h \rangle \), \( \langle f, h \rangle \), \( \langle e, i \rangle \), and \( \langle h, i \rangle \), will appear in every resulting equivalent DAGs,
which implies that they are compelled edges as can be verified by Figure 4 as well. Because the numerators \( p(hef), p(ief), \) and \( p(ebc) \) have been designated to absorb the denominators \( p(ef), p(ef), \) and \( p(bc) \), respectively, this indicates that \( hef, ief, \) and \( ebc \) have to be removed from the absorption sets \( AS(a), AS(c), \) and \( AS(e) \), if applicable. Such removal changes the absorption set for \( e \) from \( \{ebc, ge, hef, ief\} \) shown in expression 28 to the new refined singleton set \( \{ge\} \). This new refined absorption set \( AS(e) \) implies that the denominator \( p(e) \) will now have to be absorbed by the only applicable numerator \( p(ge) \) to obtain \( p(g|e) \). Therefore, the edge in \( E_{eg} \), namely, \( \langle e, g \rangle \), is also a compelled edges as can be verified by Figure 4. Recall that the denominators \( p(ef), p(ef), \) and \( p(bc) \) have been taken care of by three designated numerators for absorptions; now the denominator \( p(e) \) has been taken care of by the numerator \( p(ge) \). This indicates that \( ge \) has to be removed from the absorption set \( AS(a) \) and \( AS(c) \). However, such removal makes no change to them.

The above analysis resulted in the following expressions for the absorptions of denominators, contrasting with those in expressions 25–30:

\[
a = \{ca, da\} \tag{31}
\]

\[
bc = \{ebc\} \tag{32}
\]

\[
c = \{ca, fc\} \tag{33}
\]

\[
e = \{ge\} \tag{34}
\]

\[
ef = \{hef, ief\} \tag{35}
\]

\[
ef = \{hef, ief\} \tag{36}
\]

The compelled edges can then be identified based on the above expressions if a denominator can only be absorbed by a fixed denominator.

### 4.2. The Algorithm

Based on the above discussions, we thus propose the following algorithm to identify compelled edges in a given DAG. The correctness of the algorithm will be discussed shortly.

**ALGORITHM Identify-Compelled-Edges**

Input: a DAG.

Output: Compelled edges in the DAG stored in the set \( \text{Edges} \).

\{
1. Obtain the intrinsic factorization of the given DAG and let \( \text{Edges} = \emptyset \).
2. Let \( \textbf{D} = \{D_1, \ldots, D_m\} \) be a multiset, where \( p(D_i) \) is a denominator in the intrinsic factorization obtained in step 1.
3. For each \( D_i \in \textbf{D}, \ i = 1, \ldots, m, \) compute \( D_i \)'s absorption set \( AS(D_i) \);
   Repeat steps 4 and 5 until \( AS(D_i) \) has no change for \( i = 1, \ldots, m \).
4. For each \( D_j \in \textbf{D} \) s.t. \( |D_j| = 1, \ i = 1, \ldots, m, \)
   If \( |AS(D_j)| = 1, \)
   \[ AS(D_j) = AS(D_j) - AS(D_i), \] for all \( j \neq i, \)
5: For each $D_i \in \mathbf{D}$ s.t. $|D_i| > 1$, $i = 1, \ldots, m$,
    If $|AS(D_i)| = |D_i|$, $AS(D_j) = AS(D_j) - AS(D_i)$, for all $j \neq i$,
6: For each $D_i \in \mathbf{D}$, $i = 1, \ldots, m$,
    If $k = |AS(D_i)| = |D_i|$, $Edges = Edges \cup E_{D_i}^{Y_j}$, for each $Y_j \in AS(D_i)$ $j = 1, \ldots, k$.
7: Return Edges.
}

It is perhaps worth mentioning that steps 4 and 5 in the algorithm can be merged by removing step 4 and removing the requirement $|D_j| > 1$ from step 5 to simplify the code without altering the algorithm.\(^b\)

We now prove the correctness of the algorithm. We need to show that the algorithm captures exactly all the compelled edges in a given DAG, no more and no less. In other words, we need to prove that every edge identified by the algorithm is a compelled edge in a given DAG; every compelled edge in a given DAG can be identified by the algorithm.

**Lemma 2.** The directed edges stored in Edges in step 7 of the algorithm “Identify-Compelled-Edges” are compelled edges of the given DAG.

**Proof.** Steps 1–3 in the algorithm compute the absorption set for each denominator in the intrinsic factorization of the given DAG. These steps are obvious. In steps 4 and 5, we identify denominators in $\mathbf{D}$ that have to be absorbed by a fixed numerator and we update the absorption sets for the other denominators in $\mathbf{D}$ accordingly. The algorithm deals with two cases of such identifications and updatings in steps 4 and 5 of the algorithm.

If $D_i \in \mathbf{D}$ and $|D_i| = 1$ (step 4), in other words, $D_i$ is a distinct element in $\mathbf{D}$, consider its absorption set $AS(D_i)$. If $|AS(D_i)| = 1$, say $AS(D_i) = \{Y\}$, then $p(D_i)$ will have to be absorbed by the only applicable denominator $p(Y)$, and the CPD $p(Y - D_i | D_i)$ obtained will give rise to compelled edges in $E_{D_i}^{Y_j}$. In the meantime, because the numerator $p(Y)$ has been used to absorb $p(D_i)$, it is now not available to be used to absorb the other denominators; we thus have to update the absorption sets for all the other $AS(D_j)$, $j \neq i$, by removing $Y$ from $AS(D_j)$, which is accomplished in step 4 of the algorithm.

If $D_j \in \mathbf{D}$ and $|D_j| > 1$ (step 5), in other words, $D_j$ has multiple occurrences in $\mathbf{D}$ (because $\mathbf{D}$ is a multiset), consider its absorption set $AS(D_j)$. If $|AS(D_j)| = |D_j|$, in other words, the number of the denominator $p(D_j)$ is the same as the number of the applicable numerators that can absorb $p(D_j)$, in this case, there exists a one-to-one correspondence between each member $Y \in AS(D_j)$ and each occurrence of $p(D_j)$ as denominator such that every $p(D_j)$ will have to be absorbed by a unique $p(Y)$. Therefore, compelled edges in $E_{D_j}^{Y_j}$ (for each $Y \in AS(D_j)$) will be obtained by such absorptions. In the meantime, because all the numerators in $AS(D_j)$

\(^b\)The idea of merging steps 4 and 5 was proposed by one of the referees. We choose to leave the code as it is in order to improve the readability of the algorithm.
have been used to absorb each occurrence of the denominator \( p(D_i) \), they are now not available to be used to absorb the other denominators. We then update the absorption sets for all the other \( AS(D_j), j \neq i \), by removing elements in \( AS(D_i) \) from \( AS(D_j) \), which is accomplished in step 5 in the algorithm.

In both cases, because the denominator(or denominators) has to be absorbed by a fixed numerator (fixed numerators), the edges returned in \( Edges \) in step 7 of the algorithm are indeed the compelled edges of the given DAG.

If one takes a close look at steps 4 and 5 in the algorithm, one may question the consequences of ignoring the “else” parts of the “if” statements in both steps 4 and 5. In particular, one may be interested in knowing if compelled edges will escape from being captured by the algorithm because of the omitted “else” parts in steps 4 and 5. The following Lemma confirms that compelled edges will not escape from being captured by the algorithm.

**Lemma 3.** All the compelled edges in a given DAG can be identified by the algorithm “Identify-Compelled-Edges.”

**Proof.** We prove this lemma by contradiction. Suppose there exists a compelled edge that cannot be identified by the algorithm. According to the algorithm, there are two cases under which this situation could have happen.

1. \( |D_i| = 1 \), but \( |AS(D_i)| > 1 \). In other words, \( D_i \) is a distinct element of \( D \) whose absorption set contains more than one candidate numerators; however, \( D_i \) has to be absorbed by a particular \( Y \in AS(D_i) \) (to produce the compelled edges that cannot be identified by the algorithm), no other choices, even though other candidate numerators besides \( Y \) do exist in \( AS(D_i) \).

2. \( |D_i| > 1 \), but \( |AS(D_i)| > |D_i| \). In other words, \( D_i \) has multiple occurrences in \( D \), and the number of candidate numerators in \( AS(D_i) \) is greater than the number of \( D_i \) in \( D \); however, each \( D_i \) has to be absorbed by a particular element in \( AS(D_i) \) (to produce the compelled edges that cannot be identified by the algorithm), no other choices.

We now show both cases are impossible and lead to contradiction.

In case (1), because \( D_i \) has to be absorbed by \( Y \in AS(D_i) \), this implies that if \( Y' \in AS(D_i) \) (\( Y' \neq Y \)) is going to absorb \( D_j \); then this will cause some denominator, say \( D_j \), to not be absorbed. The only reason for \( D_j \) not being absorbed is that all the candidate numerators in \( AS(D_j) \) have been used up to absorb other denominators. There are two cases for \( D_j \), that is, \( |D_j| > 1 \) or \( |D_j| = 1 \). If \( |D_j| > 1 \), then it must be the case that \( |AS(D_j)| > |D_j| \), for otherwise, \( |AS(D_j)| = |D_j| \), and this would have been taken care of by step 5 of the algorithm, and each occurrence of \( D_j \) would have been absorbed. On the other hand, under the circumstance that \( |AS(D_j)| > |D_j| \), \( Y' \) absorbing \( D_j \) will not cause \( D_j \) to not be absorbed, even \( Y' \in AS(D_j) \). Therefore, if \( Y' \) absorbing \( D_j \) causing \( D_j \) not to be absorbed, then it must be the case that \( |D_j| = 1 \). This information implies that it would not be the case that \( AS(D_j) \) so far is a singleton set, in which case \( D_j \) would have already been absorbed by this single candidate numerator and would have been take care of by step 4 of the algorithm. This information also implies that it would not be the case that \( AS(D_j) \) so far has more than two candidate numerators, in which case \( D_j \) has plenty
of choices to be absorbed by a candidate numerator in $AS(D_j)$ after $Y'$ absorbs $D_i$ and removing $Y'$ from $AS(D_j)$ if $Y' \subseteq AS(D_j)$. Therefore, $AS(D_j)$ can have only two candidate numerators in $AS(D_j)$. Moreover, $AS(D_j) = \{Y', Z\}$ before $Y'$ absorbed $D_j$. Because $Y'$ absorbs $D_i$, $Y'$ has to be removed from $AS(D_j) = \{Y', Z\}$ such that the updated $AS(D_j) = \{Z\}$. Because $D_j$ cannot be absorbed, this implies that another denominator, say $D_k$, is competing with $D_j$ to have $Z$ as the numerator to absorb it. This indicates that $AS(D_k) = \{Y', Z\}$ as well for the same reason before $Y'$ absorbed $D_j$. Therefore, we have the situation that $AS(D_j) = \{Y', Z\}$ and $AS(D_k) = \{Y', Z\}$. A contradiction will emerge. Because $D_j$ will be absorbed by $Y$ as assumed to produce the compelled edges that cannot be identified by the algorithm, $D_j$ has to be absorbed by one of $Y'$ or $Z$, say by $Y'$, and $D_k$ has to be absorbed by $Z$. This means the following two CPDs, that is, $p(Y' - D_j | D_k)$ and $p(Z - D_k | D_k)$, will be obtained. However, in $E_{D_j}^Y$, there will exist $\langle b, c \rangle$; on the other hand, in $E_{D_k}^Z$, there will exist $\langle c, b \rangle$, where $b \in D_j - D_k$ and $c \in D_k - D_j$. Obviously, a cycle will exist in the resulting equivalent DAG and this is a contradiction.

In case (2), let $n = |D_i|$ and $m = |AS(D_i)|$; because $|AS(D_i)| > |D_i| > 1$, the occurrences of $D_i$ do not have to be absorbed by a fixed set of $n$ numerator in $AS(D_i)$, and the algorithm will not produce any compelled edges. However, if we assume that there do exist some compelled edges that escape from being captured by the algorithm, then all $n$ denominators of $D_i$ have to be absorbed by $n$ fixed candidate numerators in $AS(D_i)$. Without loss of generality, assume that all $n$ denominators $D_i$ have to be absorbed by the first $n(<m)$ candidate numerators in $AS(D_i)$, no other choices. If any one $D_i$ is absorbed by the $(n+1)$th numerator in $AS(D_i)$, then some other denominator, say $D_j$, will not be absorbed. A contradiction can be similarly reached as in case (1). This concludes the proof of the lemma.

The following theorem then follows directly based on Lemmas 2 and 3.

**Theorem 5.** The output of the algorithm “Identify-Compelled-Edges” is exactly the compelled edges in a given DAG.

### 4.3. The Complexity Analysis

The core of the algorithm is the iteration consisting of steps 4 and 5. This imposes the question of how many times the algorithm has to go through the iteration before it terminates. This is very important to the complexity of the algorithm. Note the only time that the absorption sets get updated is in steps 4 and 5 when $|AS(D_i)| = |D_i|$, which signals that compelled edges are being identified. Accompanying this identification of compelled edges, the denominators that are responsible for producing these compelled edges are known and their respective absorption sets will not be changed any more by further iterations of steps 4 and 5 because they have been removed from the absorption sets of the other denominators. For each iteration of steps 4 and 5, either some compelled edges are identified (by either step 4 or 5 or both), which incurs the updations of absorption sets, or no compelled edges are identified, which incurs no updations of absorption sets. In each iteration, if any compelled edges are identified, then the iteration
goes on to identify possibly more compelled edges in the next iteration; otherwise, if no compelled edges are identified, which means the absorption sets stay the same from the previous iteration, then this marks the end of the iteration and all compelled edges now have been identified and returned by step 7 of the algorithm. Because the compelled edges are generated by observing a denominator being uniquely absorbed by a numerator and we have at most $|D|$ denominators, it then follows that there are at most $|D|$ times of iteration of steps 4 and 5. In other words, the core of this algorithm, namely, the iteration of steps 4 and 5, will be executed at most $|D|$ times. Because set-operations can be efficiently implemented, for example, subset, intersection, and union can all be implemented in C++ using the STL with complexity $O(m + n)$, where $m$ and $n$ are sizes of two sets participating in the operations, the time complexity of the entire algorithm is dominated by $|D|$ executions of steps 4 and 5. Based on the above analysis, the iteration can be optimized as follows. Because we know that there will be at most $|D|$ executions of steps 4 and 5, we can use a “while” loop to examine every denominator in $D$. Furthermore, we can put an ordering for all the remaining denominators that have not been examined so that denominators that satisfy $|AS(D_i)| = |D_i|$ are being placed in front of those that do not satisfy. Within the “while” loop, the updating of absorption sets can be done in $O(|D|)$. Therefore, the complexity for the entire algorithm is $O(|D|)$. The fact that our algorithm is bounded by at most $|D|$ times of iteration implies a slight improvement over Chickering’s algorithm in Ref. 5. Chickering’s algorithm is a pure graphical method, and it consists of two subalgorithms. The algorithm in Ref. 5 first generates a total ordering over the edges in the given DAG; it then examines each edge according to the ordering obtained to label each edge as “compelled” or “reversible.” Chickering’s algorithm in the worst case takes time $O(|E| \cdot |U|)$, where $|E|$ is the number of edges in the given DAG, and $|U|$ is the number of nodes in the given DAG. It can be easily seen that $|D| < |E|$ and $|D| < |U|$; therefore, our algorithm is slightly better than the one in Ref 5, even though these two algorithms use different measurements of problem sizes in their respective analysis (one uses the number of edges in the given DAG, the other uses the number of denominators in the intrinsic factorization of the given DAG).

Finally, we want to point out that the idea behind our proposed algorithm suggests a different lower bound for the problem of identifying compelled edges in a DAG than the idea used in the algorithm in Ref. 5. The algorithm in Ref. 5 has to at least check each edge in a DAG once to decide if it is compelled; therefore, the lower bound for any algorithm that examines edges in a DAG requires at least $O(|E|)$ time to complete. However, in our proposed algorithm, we have to at least check each denominator in the intrinsic factorization of a given DAG to decide if compelled edges will be produced by absorbing the denominator; therefore, the lower bound for any algorithm that examines the denominators requires at least $O(|D|)$ times to complete.

5. CONCLUSION

In this article, we studied the marginal factorization of Bayesian networks. This novel perspective reveals nontrivial results that cannot be appreciated by
studying either the DAG structure or the standard Bayesian factorization (CPD factorization). In particular, we give an algebraic characterization of equivalence classes of DAGs and we suggest an algebraic method for identifying compelled edges in a DAG by utilizing the algebraic characterization of equivalent DAGs. The results in this article suggests that the study of the marginal factorization of Bayesian networks is worth being pursued further. It is perhaps worth mentioning that the algebraic characterization of equivalent DAGs was remarked in Ref. 9 and some other applications of the algebraic characterization of equivalent DAGs were recently studied in Ref. 10.

Acknowledgments

The authors thank anonymous referees for their insightful comments and critical remarks that resulted in a greatly improved revision.

References


*This information was brought to our attention by one of the referees.*