Abstraction for Efficiently Computing Most Probable Explanations in Bayesian Networks

Ole J. Mengshoel
Carnegie Mellon University
NASA Ames Research Center
Mail Stop 269-3
Moffett Field, CA 94035
Ole.J.Mengshoel@nasa.gov

Abstract

Two factors that may severly slow down computation of answers to Bayesian network queries are high graph connectivity (potentially causing high treewidth) and high node cardinalities. In this paper, where we address the problem of high node cardinalities by means of abstraction, two contributions are made. First, we formulate abstraction in Bayesian network by means of set partitioning, and make connections to previous work using abstraction hierarchies. Second, we investigate the computation of most probable explanations (MPEs) in Bayesian networks, when some nodes are abstracted. In particular, we consider the benefit of using Bayesian network abstraction when compiling Bayesian networks into join trees or arithmetic circuits, which are then used for efficient on-line computation in resource-bounded environments.

1 Introduction

Abstraction hierarchies have seen widespread use within many areas of artificial intelligence. There are several purposes of abstraction, including (i) improving human comprehension and interaction [Chang and Fung, 1991], (ii) facilitating machine learning and knowledge acquisition [Friedman and Goldszmidt, 1996; Zhang and Honavar, 2004; desJardins *et al.*, 2008], and (iii) improving the speed or quality of computation [Liu and Wellman, 2002; Sharma and Poole, 2003; Verma *et al.*, 2003].

Computing most probable explanations in Bayesian networks (BNs) [Pearl, 1988], which has application in the areas of diagnosis, image processing, and error correcting decoding, is the main focus of this paper. Exactly computing most probable explanations (MPEs) in BNs is computationally hard [Shimony, 1994], and computational difficulties arise quite regularly in complex application domains, especially when computation is severely resource-bounded in time, space, or both. We identify at least two causes of computational difficulty for discrete BNs: (i) The topology and connectedness of a BN's underlying directed acyclic graph and (ii) the high cardinality of a significant set of discrete BN nodes. Our emphasis in this paper is on this latter cause; however when these two causes co-occur they are particularly devastating. Discrete BN nodes can have high cardinalities for several reasons: First, they may represent discrete parameters, for example categorical parameters, that inherently take a large number of values [Sharma and Poole, 2003; desJardins *et al.*, 2008]. A concrete example of this might be URLs on the Web, which clearly are discrete and massive in number. A second reason for high-cardinality, discrete BN nodes is that they are used to represent continuous parameters [Liu and Wellman, 2002]. The number of states grows exponentially with the number of bits used when representing a quantized continuous parameter. Again, if a fine-grained discretization is used in BN nodes, the difficulty of computation may become a major challenge.

To attack the problem of large state spaces slowing down the speed of probabilistic computation using BNs, we consider abstraction. We focus here on the fundamentals of abstraction in BNs by carefully formulating abstraction using set partitioning, and making connections to previous work using abstraction hierarchies [Friedman and Goldszmidt, 1996; Zhang and Honavar, 2004; desJardins *et al.*, 2008]. We extend previous work on BN abstraction and refinement [Chang and Fung, 1991; Liu and Wellman, 2002; Sharma and Poole, 2003] by focusing on MPE computation. In addition, we consider the benefit of using abstraction when compiling BNs into join trees [Lauritzen and Spiegelhalter, 1988; Jensen *et al.*, 1990] or arithmetic circuits [Darwiche, 2003; Chavira and Darwiche, 2007], which are then used for efficient on-line computation in resource-bounded environments.

The remainder of this paper is organized as follows. Concepts related to Bayesian network and their approximation are first presented in Section 2. We focus on abstraction and refinement in Bayesian networks in Section 3, and in Section 4 we discuss the role of abstraction when computing MPEs for BNs. A case study is presented in Section 5, while Section 6 concludes and mentions future work.

2 Preliminaries

A Bayesian network (BN) structures a multi-variate probability distribution by using a directed acyclic graph (DAG) in which nodes represent random variables. A (discrete) BN node V is a discrete random variable with a mutually exclusive, exhaustive, and finite state space $\Omega_V = \Omega(V) = \{v_1, ..., v_m\}$. We use the notation Π_V for the parents of a

node V, Ψ_V for the children of V, and π_V for an instantiation of all parents Π_V of V. The notion of a Bayesian network can now be introduced.

Definition 1 (Bayesian network) *A Bayesian network is a tuple $\beta = (\boldsymbol{V}, \boldsymbol{E}, \boldsymbol{P})$, where $(\boldsymbol{V}, \boldsymbol{E})$ is a DAG with nodes $\boldsymbol{V} = \{V_1, ...V_n\}$, edges $\boldsymbol{E} = \{V_1, ..., V_m\}$, and where $\boldsymbol{P} = \{\Pr(V_1 \mid \Pi_{V_1}), \ldots, \Pr(V_n \mid \Pi_{V_n})\}$ is a set of conditional probability tables (CPTs). For each node $V_i \in \boldsymbol{V}$ there is one CPT, which defines a conditional probability distribution $\Pr(V_i \mid \Pi_{V_i})$.*

The independence assumptions induced by $(\boldsymbol{V},\boldsymbol{E})$ in Definition 1 imply the following joint distribution:

$$\Pr(\boldsymbol{v}) = \Pr(V_1 = v_1, \ldots, V_n = v_n) = \prod_{i=1}^{n} \Pr(v_i \mid \pi_{V_i}), \tag{1}$$

where $\Pi_{V_i} \subset \{V_{i+1}, \ldots, V_n\}$.

A BN can be provided *evidence* $\boldsymbol{e}$ by clamping evidence nodes to their respective observed states. Taking into account the evidence, different probabilistic queries can be answered using Bayesian networks. These probabilistic queries include marginals, most probable explanation (MPE), and maximum aposteriori probability (MAP). We emphasize here MPE computation, and let MPE(β, $\boldsymbol{e}$) denote the set of MPEs computed for a BN β under evidence $\boldsymbol{e}$. In order to compute MPEs efficiently and predictably in resource-bounded environments, we compile a BN β into a join tree [Lauritzen and Spiegelhalter, 1988; Jensen *et al.*, 1990] or an arithmetic circuit [Darwiche, 2003; Chavira and Darwiche, 2007] off-line. Such a compiled data structure δ is then used to compute MPE(δ, $\boldsymbol{e}$) on-line.

Abstraction and aggregation approaches have both been described in the BN literature [Chang and Fung, 1991; Liu and Wellman, 2002]. *Abstraction* is essentially to replace several node states with one node state. Abstraction is also known as state-space abstraction [Liu and Wellman, 2002], coarsening [Chang and Fung, 1991], or behavioral abstraction [Genesereth, 1984]. *Aggregation* is essentially to replace several nodes with one node. Aggregation is also known as structural abstraction [Liu and Wellman, 2002] or hierarchical abstraction.

In this paper we emphasize abstraction and now consider a classical approach [Chang and Fung, 1991]. Chang and Fung introduced the two operations of REFINE and COARSEN for discrete BNs [Chang and Fung, 1991]. COARSEN eliminates states for a node (it is an abstraction operation), while REFINE introduces new states for a node (it is a refinement operation). Both operations, which are discussed in more detail in Section 3, take as input a target node and a desired refinement or coarsening, and then output a revised conditional probability distribution for the target node and for the target node's children. Both operations are based on constraints on the Markov blanket of the target node. Two classes of operations are described: external operations and internal operations. External operations change a BN's topology while internal operations, which we emphasize in this paper, maintain the topology.

3 Abstraction in Bayesian Networks

In this section we discuss abstraction for BN nodes and BNs, and make connections to set partitions and hierarchies (DAGs that are trees) as well as previous work in the area. This section lays the foundation for Section 4, where we consider MPE computation.

3.1 Abstraction and Set Partitions

We make clear the close relationship between abstractions and set partitions. Analogous to the lattice or partial order of set partitions, there is a lattice of abstractions, and we introduce a notation to clearly state that a node $\hat{X}$'s state space $\Omega_{\hat{X}}$ abstracts the state space Ω_X of another node X.

Definition 2 (Set partitioning of node) *Consider a BN node X with state space $\Omega_X = \{x_1, \ldots, x_m\}$. Let $\mathbb{P}(\Omega_X)$ be the set partitions of Ω_X, and let P_i, $P_j \in \mathbb{P}(\Omega_X)$. We say that P_i refines P_j (or P_j abstracts P_i) and write $P_i \leq P_j$ if each block in P_i is a subset of some block in P_j.*

As an example, suppose that $\Omega_X = \{a, b, c\}$. Then the set partitions for Ω_X are $\mathbb{P}(\Omega_X) = \{\{a\}, \{b\}, \{c\}\}$, $\{\{a, b\}, \{c\}\}$, $\{\{a, c\}, \{b\}\}$, $\{\{a\}, \{b, c\}\}$, and $\{\{a, b, c\}\}$, and we have for example $\{\{a\}, \{b\}, \{c\}\} \leq \{\{a, b\}, \{c\}\}$.

There is a natural one-to-one mapping between a lattice of set partitions $\mathbb{P}$ and a lattice of BN node state spaces $\mathbb{S}$, which is closely related to an abstraction lattice $\mathbb{A}$.

Definition 3 (State space mapping) *Let P_1, $P_2 \in \mathbb{P}$ and S_1, $S_2 \in \mathbb{S}$. Let $\{b_{i,1}, \ldots, b_{i,\kappa(i)}\} = P_j \in \mathbb{P}$. Then, for each $b_{i,j} \in P_j \in \mathbb{P}$, we define*

$$f(b_{i,j}) = \begin{cases} x & \text{if } b_{i,j} = \{x\} \\ x_1 \cdots x_k & \text{if } b_{i,j} = \{x_1, \ldots, x_k\} \end{cases}$$

as well as the relation $S_1 \preceq S_2$ if an only if $P_1 \leq P_2$.

Similar to Definition 3, we introduce an abstraction lattice $\mathbb{A}$, with order $A_1 \preceq A_2$ for A_1, $A_2 \in \mathbb{A}$ that is order-isomorphic to $\mathbb{S}$ (and therefore to $\mathbb{P}$) and with a mapping g for syntactic sugaring. For example, the state space {man, woman, dog} induces a set partition $P = \{\{\text{man, woman}\}, \{\text{dog}\}\} \in \mathbb{P}$, which after abstraction is $g(f(P)) = \{\text{person, dog}\} \in \mathbb{A}$. Fundamentally, we map from the set partitions induced by the state space of a single node into all its possible abstract state spaces.

Given the abstraction lattice, we have the following definition.

Definition 4 (Node abstraction) *Let $\Omega_{\hat{X}}$, $\Omega_X \in \mathbb{A}$. If $\Omega_{\hat{X}} \succeq \Omega_X$ then we say that $\Omega_{\hat{X}}$ abstracts Ω_X (and that Ω_X refines $\Omega_{\hat{X}}$). We also use the notation $\hat{X} \succeq X$ and say that $\hat{X}$ abstracts X (and that X refines $\hat{X}$).*

For simplicity we often do not distinguish sharply between the lattices $\mathbb{P}$, $\mathbb{S}$, and $\mathbb{A}$. One exception is for the case of refinement mappings.

Definition 5 (Refinement mapping) *Let $\hat{x} \in A \in \mathbb{A}$. The refinement mapping $\rho(\hat{x})$ is defined as $\rho(\hat{x}) = f^{-1}(g^{-1}(\hat{x}))$. If $|\rho(\hat{x})| > 1$ then $\hat{x}$ is an abstracted state. If $|\rho(\hat{x})| = 1$ then $\hat{x}$ is an original state.*

Informally, a refinement mapping ρ takes an abstract state $\hat{x}$ and returns the set of its original states. For example, consider $\Omega_X = \{\text{man, woman, dog}\}$ with abstraction $\Omega_{\hat{X}} = \{\text{person, dog}\} \in \mathbb{A}$. Here, $\Omega_{\hat{X}}$ is derived from the set partition $\{\{\text{man, woman}\}, \{\text{dog}\}\} \in \mathbb{P}$ in an obvious way, giving $\rho(\text{person}) = \{\text{man, woman}\}$.

To illustrate our novel concepts, we use the internal abstraction operator for conditional probabilities [Chang and Fung, 1991]. In our variant of this scheme, the inputs to abstraction are:

- A node X whose state space Ω_X is to be abstracted to $\hat{X}$ with state space $\Omega_{\hat{X}}$: $\hat{X} \succeq X$ and $\Omega_{\hat{X}} \succeq \Omega_X$.
- A refinement mapping ρ, which describes how states $\hat{x} \in \Omega_{\hat{X}}$ are mapped into states $x \in \Omega_X$.

The outputs are the following:

- A new conditional distribution for $\hat{X}$, $\Pr'(\hat{X} \mid \Pi_{\hat{X}})$, computed from X's $\Pr(X \mid \Pi_X)$.
- New conditional distributions for successors of $\hat{X}$, $\Pr'(S_{\hat{X}} \mid \Pi_{S_{\hat{X}}})$, where $S_{\hat{X}}$ is a successor (child) of $\hat{X}$, $S_{\hat{X}} \in \Psi_{\hat{X}}$.

In computing these new conditional distributions, the central idea of Chang and Fung is to keep the effect of abstraction and refinement localized by maintaining the joint distribution of the Markov blanket of the abstracted node intact, if possible. Their idea leads to the following two types of constraints on abstraction.

Definition 6 *Let $\Omega_X \preceq \Omega_{\hat{X}}$ and suppose that $x \in \Omega_X$, $\hat{x} \in \Omega_{\hat{X}}$, and $x \in \rho(\hat{x})$. The* node constraint *is*

$$\Pr(\hat{x} \mid \pi_X) = \sum_{x \in \rho(\hat{x})} \Pr(x \mid \pi_X)). \tag{2}$$

Let $s_X \in \Omega_{S_X}$, with $S_X \in \Psi_X$, and assume $x \in \pi_{S_X}$ and $\hat{x} \in \hat{\pi}_{S_X}$. The child constraint *is*

$$\Pr(s_X \mid \hat{\pi}_{S_X}) \Pr(\hat{x} \mid \pi_X) = \sum_{x \in \rho(\hat{x})} \Pr(s_X \mid \pi_{S_X}) \Pr(x \mid \pi_X). \tag{3}$$

The parent and child constraints can be used to compute CPTs for the child of a node with abstracted state $\hat{x}$, using the following form of (3):

$$\Pr(s_X \mid \hat{\pi}_{S_X}) = \frac{\sum_{x \in \rho(\hat{x})} \Pr(s_X \mid \pi_{S_X}) \Pr(x \mid \pi_X)}{\Pr(\hat{x} \mid \pi_X)}. \tag{4}$$

We can now introduce abstraction relations between BNs, similar to for nodes and state spaces.

Definition 7 (Bayesian network abstraction) *Consider BNs $\beta_1 = (\boldsymbol{V}_1, \boldsymbol{E}_1, \boldsymbol{P}_1)$ and $\beta_2 = (\boldsymbol{V}_2, \boldsymbol{E}_2, \boldsymbol{P}_2)$. If $\Omega(V_{1,i}) \succeq \Omega(V_{2,i})$ for all $V_{1,i} \in \boldsymbol{V}_1$ and $V_{2,i} \in \boldsymbol{V}_2$, then β_1 abstracts β_2, which we write $\beta_1 \succeq \beta_2$. If $\Omega(V_{1,i}) \preceq \Omega(V_{2,i})$ for all $V_{1,i} \in \boldsymbol{V}_1$ and $V_{2,i} \in \boldsymbol{V}_2$, then β_1 refines β_2, which we write $\beta_1 \preceq \beta_2$. If neither $\beta_1 \succeq \beta_2$ nor $\beta_1 \preceq \beta_2$, then β_1 and β_2 are incomparable, and we write $\beta_1 \parallel \beta_2$ (or $\beta_2 \parallel \beta_1$).*

Given an abstracted Bayesian network $\beta_A \succeq \beta_R$, we now consider the situation where an explanation $\boldsymbol{x}_A$ in β_A abstracts another explanation $\boldsymbol{x}_R$ in β_R, so these explanations are closely related, for example through the use of one or more abstraction hierarchies.

Definition 8 (Explanation abstraction) *Consider BNs $\beta_A = (\boldsymbol{V}_A, \boldsymbol{E}_A, \boldsymbol{P}_A)$ and $\beta_R = (\boldsymbol{V}_R, \boldsymbol{E}_R, \boldsymbol{P}_R)$ where $\beta_A \succeq \beta_R$. Let $\boldsymbol{x}_A$ be an explanation in β_A and let $\boldsymbol{x}_R$ be an explanation in β_R. Then $\boldsymbol{x}_A$ abstracts $\boldsymbol{x}_R$, which we write $\boldsymbol{x}_A \succeq \boldsymbol{x}_R$, and $\boldsymbol{x}_R$ refines $\boldsymbol{x}_A$, which we write $\boldsymbol{x}_R \preceq \boldsymbol{x}_A$.*

Suppose we have a BN β_A which is an abstraction of another BN β_R, or $\beta_A \succeq \beta_R$. From the point of view of fault diagnosis using BNs, there are several reasons why β_A may be of interest, despite the fact that β_R may be a more accurate physical model. (i) First, β_A can in some cases be an end-result in itself; the BN β_R may contain distinctions that are irrelevant to the task at hand. For example, suppose that β_R contains a wide range of fault types, but we are interested in fault detection rather than a detailed fault diagnosis. In this case, one way want to construct an β_A, with $\beta_A \succeq \beta_R$, in which system health nodes only have two states, say "healthy" and "faulty", and all fault types are abstracted into the latter state. (ii) Second, β_A (or δ_A) can be considered to be a stepping-stone for computation with β_R (or δ_R). In this case, one may be willing to use β_A perhaps even in the face of potential loss in accuracy, because inference is faster in β_A than in β_R. That said, it is clear that abstraction approaches that are guaranteed to or typically produce "high-quality" results are of greatest interest. For instance, we would like an MPE in β_R to abstract to an explanation that is an MPE in β_A, and vice versa.

3.2 Abstraction and Hierarchies

An abstraction hierarchy is a DAG, typically a tree, which structures the state space of a BN node. In the following we partition the nodes $\boldsymbol{V}$ in a DAG into leaf nodes $\boldsymbol{L}$, root nodes $\boldsymbol{R}$, and trunk nodes $\boldsymbol{T}$.

Definition 9 (Abstraction hierarchy) *Let X be a BN node with $\Omega_X = \{x_1, ..., x_m\}$. An abstraction hierarchy $\boldsymbol{H} = (\boldsymbol{V}, \boldsymbol{E})$ for X is a DAG that is a tree with exactly one root node $R \in \boldsymbol{R}$, where each non-leaf node has at least two children, and with a one-to-one mapping $\ell : \Omega_X \rightarrow \boldsymbol{L}$, where $\boldsymbol{L} \subseteq \boldsymbol{V}$ are leaf nodes.*

Abstraction hierarchies are also known as decision trees, attribute value hierarchies [desJardins *et al.*, 2008], and attribute value taxonomies [Zhang and Honavar, 2004].

We now select nodes from an abstraction hierarchy $\boldsymbol{H} = (\boldsymbol{V}, \boldsymbol{E})$ in order to form a new, abstract state space from an existing state space as represented by leaf nodes in $\boldsymbol{H}$. Note that an arbitrary set of nodes $\boldsymbol{W} \subseteq \boldsymbol{V}$ might or might not be a valid (abstracted) state space. In an abstraction hierarchy $\boldsymbol{H} = (\boldsymbol{V}, \boldsymbol{E})$ we say that a leaf node $L \in \boldsymbol{L} \subseteq \boldsymbol{V}$ is covered once by some node $W \in \boldsymbol{W} \subseteq \boldsymbol{V}$ if $L = W$. In addition, L is covered once for each directed path from W to L. The number of times that L is covered in $\boldsymbol{W}$ is given by the total number times each node $W \in \boldsymbol{W}$ covers L.

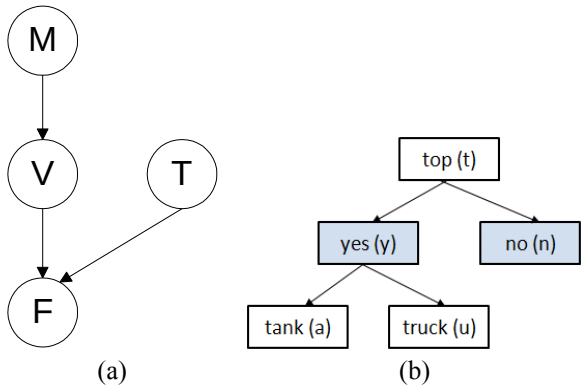

Figure 1: Bayesian network (a) with abstraction hierarchy (b) for the node V. For the abstraction hierarchy, a valid abstraction is indicated by shading.

Definition 10 (Valid abstraction) *Let $\boldsymbol{H} = (\boldsymbol{V}, \boldsymbol{E})$ be an abstraction hierarchy. Let $\boldsymbol{L}' \subseteq \boldsymbol{L}$, $\boldsymbol{T}' \subseteq \boldsymbol{T}$ and $\boldsymbol{A} \subseteq \boldsymbol{V}$. We define $\boldsymbol{A}$ as a valid abstraction for $\boldsymbol{H}$ if each $L \in \boldsymbol{L}$ is covered exactly once by $\boldsymbol{A}$.*

Figure 1 shows an example BN and abstraction hierarchy (see [Chang and Fung, 1991]). Here, M stands for military unit type, with $\Omega_M = \{b, c\}$. V stands for a vehicle in a particular location, with $\Omega_V = \{a, u, n\}$ (tank, truck, or no). T stands for terrain conditions, with $\Omega_T = \{g, b\}$ (good or bad). F is an observable feature, $\Omega_F = \{a, b, o\}$. Conditional probability tables for this example are shown in Table 1 and Table 2. Here, V's state space can be abstracted from $\Omega_V = \{a, u, n\}$ to $\Omega_{\hat{V}} = \{y, n\}$, so $\Omega_V \preceq \Omega_{\hat{V}}$ and $\rho(\hat{x}) = \rho(y) = \{a, u\}$. These two valid abstractions are represented in the abstraction hierarchy in Figure 1.

3.3 From Hierarchies to Set Partitions

To map from abstraction hierarchies to abstraction set partitions, we introduce the algorithm H2SP($\boldsymbol{H}, \boldsymbol{A}$), which takes as input an abstraction hierarchy $\boldsymbol{H}$ and a valid abstraction $\boldsymbol{A}$ for $\boldsymbol{H}$, and which outputs a set $\boldsymbol{S}$:

1. $\boldsymbol{S} \leftarrow \{\}$
2. For each $A \in \boldsymbol{A}$
   (a) If A is a leaf in $\boldsymbol{H}$ then $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{\{A\}\}$
   (b) Else $\boldsymbol{S} \leftarrow \boldsymbol{S} \cup \{\boldsymbol{L}'\}$, where $\boldsymbol{L}'$ is the set of leaf nodes reachable from A in $\boldsymbol{H}$
3. Return $\boldsymbol{S}$.

We have the following result for H2SP, where the proof is omitted due to space restrictions.

Theorem 11 *Let $\boldsymbol{H}$ be an abstraction hierarchy and let $\boldsymbol{A}$ be a valid abstraction in $\boldsymbol{H}$. Then $\boldsymbol{S}$, returned by H2SP, is a set partition over the leaf nodes in $\boldsymbol{H}$.*

In other words, if we consider the set $\{\boldsymbol{A} \mid \boldsymbol{A}$ is a valid abstraction set for $\boldsymbol{H}\}$, it is a subset of all set partitions for the leaf nodes in $\boldsymbol{H}$.

4 Computing MPEs using Abstraction

Having introduced concepts for abstraction and refinement, we now turn to how they can be utilized when computing MPEs in BNs. Due to our interest in resource-bounded computation, we carefully split the different computational tasks into off-line and on-line phases.

4.1 Off-Line Computation

The purpose of off-line computation is to create a BN that can be compiled, by different compilation algorithms, into data structures that are optimized for on-line computation.

Creating abstractions: So far, we have assumed that abstraction hierarchies exist; we now briefly discuss their genesis. In some domains there are natural constraints that can be encoded in abstraction hierarchies. An example of a domain where a predefined abstraction hierarchy is natural is in the discretization of continuous state spaces [Liu and Wellman, 2002]. Abstraction hierarchies can also be provided by experts [Zhang and Honavar, 2004; desJardins *et al.*, 2008] or created using machine learning approaches like clustering [Zhang and Honavar, 2004; desJardins *et al.*, 2008].

Using abstractions: Abstraction hierarchies are created off-line, and then used off-line to create BNs with valid abstractions in polynomial time, by iterating over all the nodes in a BN, using the appropriate abstraction and refinement operators discussed above. We create β_A from another BN β_R using abstraction concepts, compile β_A into an on-line data structure δ_A, and then use δ_A, either a join tree or an arithmetic circuit, in the on-line algorithm below.

4.2 On-Line Computation

For decision making purposes, the most important question is whether a system computes the correct MPE, and whether the probability is correct is not always equally important. This observation is utilized in the following novel algorithm called AbsMPE(β_R, δ_A, $\boldsymbol{e}$). For the time being, we assume that evidence nodes are not abstracted, so the evidence is treated exactly the same in β_A, δ_A, β_R, and δ_R.

1. Compute MPEs in δ_A, given evidence $\boldsymbol{e}$: MPE(δ_A, $\boldsymbol{e}$) $= \boldsymbol{X}^* = \{\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_m^*\}$.
2. Compute, for each $\boldsymbol{x}_i^* \in \boldsymbol{X}^*$, explanations $\{\boldsymbol{y}_{i,1}, \ldots, \boldsymbol{y}_{i,\kappa(i)}\} = \{\boldsymbol{y} \mid \boldsymbol{y}$ is an explanation in β_R and $\boldsymbol{y} \preceq \boldsymbol{x}_i^*\}$.
3. Let $\boldsymbol{Y} = \{\boldsymbol{y}_{1,1}, \ldots, \boldsymbol{y}_{1,\kappa(1)}\} \cup \cdots \cup \{\boldsymbol{y}_{i,1}, \ldots, \boldsymbol{y}_{i,\kappa(i)}\} \cup \cdots \cup \{\boldsymbol{y}_{m,1}, \ldots, \boldsymbol{y}_{m,\kappa(m)}\}$.
4. Form $\boldsymbol{Y}' \subseteq \boldsymbol{Y}$, where $\boldsymbol{Y}' = \{\boldsymbol{y} \in \boldsymbol{Y} \mid \Pr(\boldsymbol{y} \mid \boldsymbol{e}) \geq \Pr(\boldsymbol{z} \mid \boldsymbol{e})$ for all $\boldsymbol{z} \in \boldsymbol{Y}\}$.
5. Return $\boldsymbol{Y}'$.

Under certain assumptions, we have $\boldsymbol{Y}' = \boldsymbol{Y}^*$, where $\boldsymbol{Y}^* = \text{MPE}(\beta_R, \boldsymbol{e})$. Formally, we have this result.

Theorem 12 *Let β_A and β_R be two BNs such that $\beta_A \succeq \beta_R$, β_A is complied to δ_A, and assume evidence $\boldsymbol{e}$. Let $\boldsymbol{Y}^* =$ MPE(β_R, $\boldsymbol{e}$), and let $\boldsymbol{Y}' =$ AbsMPE(β_R, δ_A, $\boldsymbol{e}$). If any $\boldsymbol{y}^* \in \boldsymbol{Y}^*$ abstract into some $\boldsymbol{x}^* \in \boldsymbol{X}^*$ (abstraction assumption) then $\boldsymbol{Y}^* = \boldsymbol{Y}'$.*

$\Pr(F \mid V,T)$						
V:	a		u		n	
T:	g	b	g	b	g	b
a	**0.4**	*0.1*	**0.6**	*0.4*	0.1	0.2
b	**0.5**	*0.5*	**0.3**	*0.2*	0.1	0.2
o	**0.1**	*0.4*	**0.1**	*0.4*	0.8	0.6

$\Pr'(F \mid \hat{V},T)$				
$\hat{V}$:	y		n	
T:	g	b	g	b
a	**0.5375**	*0.30625*	0.10	0.20
b	**0.3625**	*0.29375*	0.10	0.20
o	**0.1000**	*0.40000*	0.80	0.60

Table 1: At the top, *F*'s conditional probability table is shown before abstraction of $\Omega_V = \{a, u, n\}$ to $\Omega_{\hat{V}} = \{y, n\}$, at the bottom it is shown after abstraction.

$\Pr(V \mid M)$		
M:	b	c
a	**0.3**	*0.1*
u	**0.5**	*0.3*
n	**0.2**	*0.6*

$\Pr'(\hat{V} \mid M)$		
M:	b	c
y	**0.8**	*0.4*
n	**0.2**	*0.6*

Table 2: To the left *V*'s CPT is shown before abstraction of $\Omega_V = \{a, u, n\}$ to $\Omega_{\hat{V}} = \{y, n\}$, to the right it is shown after abstraction.

Proof. To prove $\boldsymbol{Y}^* \subseteq \boldsymbol{Y}'$, we suppose that there was some $\boldsymbol{y}^* \in \boldsymbol{Y}^*$ that has an abstraction $\boldsymbol{x}^* \in \boldsymbol{X}^*$, but $\boldsymbol{y}^* \notin \boldsymbol{Y}'$. Clearly, this cannot happen since an MPE in β_R obviously will be part of $\boldsymbol{Y}'$. To prove $\boldsymbol{Y}' \subseteq \boldsymbol{Y}^*$, we suppose that there was some $\boldsymbol{y}' \in \boldsymbol{Y}'$, that has an abstraction $\boldsymbol{x}^* \in \boldsymbol{X}^*$, but $\boldsymbol{y}' \notin \boldsymbol{Y}^*$. In AbsMPE, $\boldsymbol{Y}$ is formed by taking refined explanations $\boldsymbol{y} \preceq \boldsymbol{x}^*$ for all $\boldsymbol{x}^* \in \boldsymbol{X}^*$. In particular, all $\boldsymbol{y}^*$ will be taken by the abstraction assumption, thus there is in $\boldsymbol{Y}'$ at least one $\boldsymbol{y}_1^* \in \boldsymbol{Y}^*$. Suppose that there exists some $\boldsymbol{y}' \in \boldsymbol{Y}^{'}$ such that $\boldsymbol{y}' \notin \boldsymbol{Y}^*$. By construction of AbsMPE, we then have $\Pr(\boldsymbol{y}' \mid \boldsymbol{e}) \geq \Pr(y_1^* \mid \boldsymbol{e})$ which is impossible, concluding our proof. ■

Roughly speaking, the set of MPEs is the same whether it is computed directly from β_R or from δ_A and β_R using the steps of AbsMPE presented above.

Our work is different from related work that integrates abstraction and probabilistic computation [Liu and Wellman, 2002; Sharma and Poole, 2003; Verma *et al.*, 2003] in a number of ways. Most notably, we compute MPEs, not marginals, and we perform abstraction and refinement off-line, not on-line. In contrast, Sharma and Poole compute marginals using variable elimination and do abstraction as part of inference. Liu and Wellman perform abstraction-based iterative refinement on-line; they also compute marginals.

4.3 Discussion and Analysis

Using a traditional approach, we compile off-line β_R into a data structure δ_R for which MPE(δ_R, $\boldsymbol{e}$) is computed on-line. In the AbsMPE(β_R, δ_A, $\boldsymbol{e}$) approach, on the other hand, β_A is compiled off-line into a compiled data structure δ_A, while β_R is kept as a Bayesian network. With those two approaches in mind, the AbsMPE approach may be beneficial when one or more of the following conditions hold:

- The size of the compiled structure δ_A is very small compared to the size of δ_R. For example, one or a few cliques may contribute a large part of the total clique tree size for the clique tree δ_R of β_R, and if a few nodes in those cliques can be abstracted, then that will reduce the clique tree size substantially.
- The number of MPEs in β_A and δ_A is relatively small, such that the number of subsets used to form $\boldsymbol{Y}$ in AbsMPE is not too large.
- The number of abstracted states in β_A and δ_A is relatively small, such that the size of each subset used to form $\boldsymbol{Y}$ in AbsMPE is not too large.

We now consider in more detail the size of δ_A versus that of δ_R. Suppose that a clique $\gamma \in \delta_R$ contains nodes $\{V_1, ..., V_n\}$. Then the cardinality of γ is:

$$|\gamma| = |\Omega_{V_1}| \times \cdots \times |\Omega_{V_n}|,$$

while the cardinality of a corresponding clique $\hat{\gamma} \in \delta_A$ compiled from abstracted BN β_A is

$$|\hat{\gamma}| = |\Omega_{\hat{V}_1}| \times \cdots \times |\Omega_{\hat{V}_n}|,$$

where $\hat{V}_i$ in β_A is the abstracted node of node V_i in the original BN β_R. If a clique contains ten nodes, each with ten states, and all abstracted nodes contain seven states, this yields a ratio of $|\hat{\gamma}|/|\gamma| = 2.824\,8 \times 10^{-2}$. This represents a substantial reduction in clique size, and hence in requirement for storage and computation time, especially in light of the moderate reduction in state space per node.

5 Case Study

As a case study, we compute probability values in two closely related cases for Figure 1, leading to an abstracted CPT parameter $\Pr'(a \mid u, g) = 0.5375$ as shown in Table 1. Case (i) and Case (ii) are for the abstraction hierarchy shown in Figure 1. Case (i): Let $M = b$ and consider node instantiations $\hat{V} = y$, $F = a$, and $T = g$. Using (4) we then get:

$$\frac{\sum_{x \in \rho(y)} \Pr(a \mid x, g) \Pr(x \mid b)}{\Pr(y \mid b)} = 0.525. \tag{5}$$

Case (ii): Let $M = c$, with the other nodes instantiated as above (i.e. $\hat{V} = y$, $F = a$, and $T = g$):

$$\frac{\sum_{x \in \rho(y)} \Pr(a \mid x, g) \Pr(x \mid c)}{\Pr(y \mid c)} = 0.55. \tag{6}$$

Now take the average of the values computed in Equation 5 and Equation 6; this gives the abstracted value

$$\Pr'(F = a \mid \hat{V} = y, T = g) = \frac{0.525 + 0.55}{2} = 0.5375 \tag{7}$$

$\boldsymbol{e}$	MPE(δ_R,$\boldsymbol{e}$)			MPE(δ_A,$\boldsymbol{e}$)		
F	M	V	T	M	$\hat{V}$	T
a	b	u	g	b	y	g
b	b	a	g	b	y	g
	b	u	g			
o	c	n	g	c	n	g

Table 3: The MPEs for different values of the evidence variable F for two different compiled BNs δ_R and δ_A. Here, δ_A is an abstraction of δ_R in that node $\hat{V}$ (in β_A) is an abstraction of V (in β_R).

as shown in the CPT at the bottom of Table 1, in the column with $V = y$ and $T = g$, and the row with $F = a$. The other CPT parameters can be computed in a similar way, see the highlighted parameters of $\Pr'(F \mid \hat{V}, T)$ in Table 1.

Results for our case study are shown in Table 3. We consider the row for $F = a$ in detail. In this case, we obtain $\boldsymbol{Y}^*$ = MPE(β_R, $\{F = a\}$) = $\{\{M = b, V = u, T = g\}\}$. Since MPE($\beta_A$, $\{F = a\}$) = $\{\{M = b, \hat{V} = y, T = g\}\}$, we have in AbsMPE (because $\hat{V}.y \succeq V.u$ and $\hat{V}.y \succeq V.a$) that $\boldsymbol{Y}$ = $\{\{M = b, V = u, T = g\}, \{M = b, V = a, T = g\}\}$. Since $\Pr(M = b, V = u, T = g, F = a) = 0.09$ and $\Pr(M = b, V = a, T = g, F = a) = 0.036$, AbsMPE returns $\boldsymbol{Y}' = \{\{M = b, V = u, T = g\}\}$. Hence, we have $\boldsymbol{Y}^*$ = $\boldsymbol{Y}'$ as desired.

Total clique tree size for δ_A is $|\hat{\gamma}| = 16$, since clique $\{M, \hat{V}\}$ has size 4 and clique $\{\hat{V}, F, T\}$ has size 12. The total clique tree size for δ_R is $|\gamma| = 24$, since clique $\{M, V\}$ has size 6 and clique $\{V, F, T\}$ has size 18.

While restricted to a small case study, this is an example of successful abstraction for MPE computation, since the results obtained by using AbsMPE on a smaller, abstracted clique tree δ_A (size 16) along with the refined BN β_R were exactly the same as those obtained by computing MPEs directly on the clique tree δ_R of the refined BN β_R (size 24).

6 Conclusion

Bayesian networks are seeing increased use in artificial intelligence and other sciences and disciplines. However, Bayesian network computation is also inherently computationally hard for most interesting inference tasks. This is the case, for instance, for the problem of computing most probable explanations (MPEs) in Bayesian networks.

Varying the size of the state spaces of discrete BN nodes can have an impact of several orders of magnitude on the speed of inference. Constructing abstracted BNs based on the use of abstraction hierarchies is consequently one promising approach to improve computational speed by BN approximation. State space abstraction for BNs has been investigated earlier, however this among the first times abstraction has been used in the context of computing an MPE in a Bayesian network. Specifically, we have made a connection between abstraction using set partitioning and abstraction using hierarchies or taxonomies. In addition, we have emphasized the benefit of using abstraction when compiling BNs into join trees [Lauritzen and Spiegelhalter, 1988; Jensen *et al.*, 1990] or arithmetic circuits [Darwiche, 2003; Chavira and Darwiche, 2007]. In future work, we plan to experiment with larger BNs as well as validate the potential of the approach for on-line computation in resource-bounded environments.

References

[Chang and Fung, 1991] K. Chang and R. Fung. Refinement and coarsening of Bayesian networks. In *Uncertainty in Artificial Intelligence 6*, pages 435–445. 1991.

[Chavira and Darwiche, 2007] M. Chavira and A. Darwiche. Compiling Bayesian networks using variable elimination. In *IJCAI-07*, pages 2443–2449, 2007.

[Darwiche, 2003] A. Darwiche. A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50(3):280–305, 2003.

[desJardins *et al.*, 2008] M. desJardins, P. Rathod, and L. Getoor. Learning structured Bayesian network: Combining abstraction hierarchies and tree-structured conditional probability tables. *Computational Intelligence*, 24(1):1–22, 2008.

[Friedman and Goldszmidt, 1996] N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In *UAI-96*, pages 252–262, 1996.

[Genesereth, 1984] M. Genesereth. The use of design descriptions in automated diagnosis. *Artificial Intelligence*, 24:411–436, 1984.

[Jensen *et al.*, 1990] F. V. Jensen, K. G. Olesen, and S. K. Andersen. An algebra of Bayesian belief universes for knowledge-based systems. *Networks*, 20(5):637–659, August 1990.

[Lauritzen and Spiegelhalter, 1988] S. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *Journal of the Royal Statistical Society series B*, 50(2):157–224, 1988.

[Liu and Wellman, 2002] C. Liu and M. P. Wellman. Evaluation of Bayesian networks with flexible state-space abstraction methods. *International Journal of Approximate Reasoning*, 30:1–39, 2002.

[Pearl, 1988] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[Sharma and Poole, 2003] R. Sharma and D. Poole. Efficient inference in large discrete domains. In *UAI-03*, pages 535–542, 2003.

[Shimony, 1994] E. Shimony. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68:399–410, 1994.

[Verma *et al.*, 2003] V. Verma, S. Thrun, and R. Simmons. Variable resolution particle filter. In *IJCAI-03*, 2003.

[Zhang and Honavar, 2004] J. Zhang and V. Honavar. AVT-NBL: an algorithm for learning compact and accurate naive Bayes classifiers from attribute value taxonomies and data. In *ICDM-04*, pages 289–296, 2004.