Probabilistic Reasoning with Hierarchically Structured Variables

Rita Sharma
Department of Computer Science
University of British Columbia
http://www.cs.ubc.ca/spider/rsharma/
rsharma@cs.ubc.ca

David Poole
Department of Computer Science
University of British Columbia
http://www.cs.ubc.ca/spider/poole/
poole@cs.ubc.ca

Abstract

Many practical problems have random variables with a large number of values that can be hierarchically structured into an abstraction tree of classes. This paper considers how to represent and exploit hierarchical structure in probabilistic reasoning. We represent the distribution for such variables by specifying, for each class, the probability distribution over its immediate subclasses. We represent the conditional probability distribution of any variable conditioned on hierarchical variables using *inheritance*. We present an approach for reasoning in Bayesian networks with hierarchically structured variables that dynamically constructs a *flat Bayesian network*, given some evidence and a query, by collapsing the hierarchies to include only those values necessary to answer the query. This can be done with a single pass over the network. We can answer the query from the flat Bayesian network using any standard probabilistic inference algorithm such as variable elimination or stochastic simulation. The domain size of the variables in the flat Bayesian network is independent of the size of the hierarchies; it depends on how many of the classes in the hierarchies are directly associated with the evidence and query. Thus, the representation is applicable even when the hierarchy is conceptually infinite.

1 Introduction

Many problem domains have discrete variables with a large number of values that can be represented *a priori* as an *abstraction* hierarchy (or an *is-a* hierarchy or *taxonomic* hierarchy) [Pearl, 1986; Mackworth *et al.*, 1985]. We call these variables *hierarchical variables*. Taxonomic hierarchies allow us to manage effectively the large state space of a variable because they allow facts and regularities to be revealed both at high and low levels of abstraction. In taxonomic hierarchies the information from high-level abstractions are automatically inherited by more specific concepts. Abstraction hierarchies also allow us to answer more abstract queries. As an example of a hierarchical variable, consider a random variable LT that describes the species of living things. The large number of values of LT (i.e., millions of species) can be classified according to Linnaean taxonomy hierarchy.[1] Living things are divided into kingdoms (e.g., *plantae*, *animalia*), classes (e.g., mammals, birds, fish), all the way down to species.

In this paper, we look at two related problems with hierarchical variables in Bayesian networks. The first is the compact representation of conditional probability distributions. The second is how to exploit that representation in probabilistic inference for computational gain. We assume that the tree hierarchy of the values is fixed and does not change with the context.

To compute a posterior probability, given some evidence in a Bayesian network that has both simple and hierarchical variables, we construct a flat Bayesian network by collapsing the hierarchies and including only those values necessary to answer the query. We can answer the query from the flat Bayesian network using any standard inference algorithm. The domain size of the variables in the flat Bayesian network is independent of the size of the hierarchies; it depends on how many of the classes in the hierarchy are directly associated with the evidence and query.

2 Hierarchical Variables

We divide the discrete random variables into two categories: **simple variables** and **hierarchical variables**. We call Bayesian networks that have both simple and hierarchical variables *hierarchical Bayesian networks*.[2] We use upper case letters to denote simple random variables (e.g., $X1$, $X2$, X) and the actual value of these variables by small letters (e.g., a, b, x_1). The domain of X, written $Val(X)$, is the set of values that X can take on.

A hierarchical variable is a variable in which subsets of the values of the variable are represented hierarchically in a tree [Pearl, 1986]. We refer to a node in the tree as a *class*. The nodes in the tree represent subsets of the values of the variable. The root of the tree represents the set of all the values of the variable. The children of a node correspond to a partitioning of the set of values of the node. Thus, the subsets represented by the children of a class must be mutually disjoint and any class represents the union of its children in the tree.

[1] *http* : *//www.wikipedia.org/wiki/Linnaean_taxonomy*

[2] The term hierarchical Bayesian network used in this paper is different than the term hierarchical Bayesian model used in statistics.

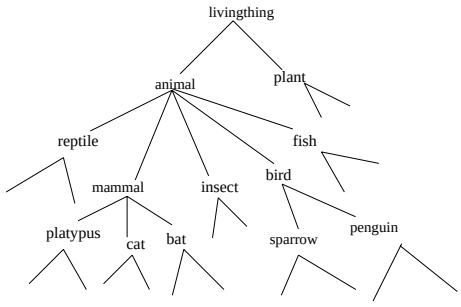

Figure 1: Part of a taxonomic hierarchy of living things

If a class C_j is a child of C_i, we say that C_j is an *immediate subclass* of C_i. The *subclass* relation is the symmetric transitive closure of the immediate subclass relation. That is, C_j is a subclass of C_i, written $C_j \leq C_i$, if C_j is a descendant of C_i in the abstraction hierarchy (or is C_i). The inverse of subclass is *superclass*. The conjunction of two classes C_j and C_k, written as $C_j \wedge C_k$, denotes the set intersection of C_j and C_k. This is empty unless one is a superclass of the other.

We denote the hierarchical variables using boldface uppercase letters (e.g., **LT**). For example, a part of the tree hierarchy of values of variable **LT** is shown in Figure 1. *tree*(**LT**) denotes the tree hierarchy of **LT**.

3 Conditional Probabilities of Hierarchical Bayesian Networks

There are two main issues to be considered to represent hierarchical Bayesian networks:

C1: specifying the conditional probability for hierarchical variables;

C2: specifying how variables are conditioned on hierarchical parents.

The simplest case of **C1** is how to specify the prior probability of a hierarchical variable. The simplest case of **C2** is how to specify the conditional probability of a simple variable conditioned on a hierarchical variable. The more complicated cases are built from these simpler cases.

3.1 Prior Probability for Hierarchical Variables

The prior probability of a hierarchical variable can be specified in a top-down manner. For each class that has subclasses, we specify a probability distribution over its immediate subclasses. For example, suppose we want to specify the prior over the hierarchical variable **LT** as shown in Figure 1. The root of the hierarchy is the class *livingthing*. We specify the probability distribution over its immediate subclasses:

$$P(animal|livingthing) = 0.4$$
$$P(plant|livingthing) = 0.6$$

We can specify the probability distribution over the immediate subclasses of *animal*. Suppose we have as part of this distribution:

$$P(mammal|animal) = 0.3$$
$$P(bird|animal) = 0.2, \text{ etc.}$$

We can specify the probability of an immediate subclass of *mammal* given *mammal*, with probabilities such as:

$$P(bat|mammal) = 0.1$$

We can compute the prior probability of any class in a recursive manner by multiplying the probability of class given its immediate superclass and the probability of its immediate superclass. The root has probability 1 (as it represents the set of all values). For example, given the probabilities as above, $P(bat)$ can be computed as follows:

$$\begin{aligned} P(bat) &= P(bat|mammal) \times P(mammal|animal) \times \\ &\quad P(animal|livingthing) \\ &= 0.012 \end{aligned}$$

In this representation, computing the probability of any class is linear in the depth of the class in the hierarchy and otherwise is not a function of the hierarchy's size.

3.2 Hierarchical Variable Is the Parent of a Simple Variable

Suppose that **H** is a hierarchical variable, F is a simple variable, and **H** is the only parent of F. We can exploit any hierarchical structure by assuming that the influence on F is local in the hierarchy of **H**. To specify the conditional probability distribution (CPD) $P(F|\mathbf{H})$ we use the notion of a *default probability distribution*. The default probability distribution of F for class C_j, written $P_d(F|C_j)$, is assumed to be inherited by all subclasses of C_j, unless overridden by a more specific default probability distribution. We can represent $P(F|\mathbf{H})$ by specifying $P_d(F|C_j)$ for some classes C_j in the hierarchy of **H**. We call such classes C_j the *exceptional classes* of **H**. The idea of using "inheritance" with abstraction hierarchies is similar to the use of "context specific independence" for the compact representation of the CPTs in a Bayesian network [Boutilier *et al.*, 1996].

To make this properly defined, we need to ensure that every terminal node in the tree hierarchy of **H** is covered by some default probability distribution. The union of all of the sets of values associated with the exceptional classes must equal $Val(\mathbf{H})$. This condition holds trivially if the root class is an exceptional class for F. If the root class is not an exceptional class, there must be an exceptional class along every path from the root.

The conditional probability of F for any class C_k of **H**, $P(F|C_k)$, can be computed as follows:

- if C_k does not have any exceptional subclasses, $P(F|C_k)$ is inherited from the default probability distribution of F for the lowest exceptional superclass C_j of C_k. Thus,
$$P(F|C_k) = P_d(F|C_j)$$
- otherwise, $P(F|C_k)$ can be derived from its children
$$P(F|C_k) = \sum_{\forall C_i \in children(C_k)} P(F|C_i) \times P(C_i|C_k)$$

Example: Consider a Bayesian network in which a hierarchical variable **LT** is the only parent of a simple variable *FLYING* with domain $\{flying, \neg flying\}$. To represent $P(FLYING|\mathbf{LT})$,

we need to define the default probability distributions over *FLYING* for the exceptional classes of **LT**. For example, we can state that *livingthings* have a low probability of *flying* by default, but *bird* and *insect* are exceptional because they have a high probability of flying. From the children of class *mammal*, a *bat* is exceptional because it has a high probability of flying. From the children of class *bird*, a *penguin* is exceptional because it has a very low probability of flying. Thus, to represent $P(FLYING|\mathbf{LT})$ we could have:

$P_d(flying|livingthing) = 0.00001$
$P_d(flying|bird) = 0.5$
$P_d(flying|bat) = 0.3$
$P_d(flying|penguin) = 0.000001$
$P_d(flying|insect) = 0.4$

Note that $P(flying|livingthing) \neq 0.00001$ as the class *livingthing* contains *bird* and *bat* that each have a much higher probability of flying.

3.3 The General Case

The general case is when a random variable (hierarchical or simple) can have any number and any kind of variables as parents. To represent the CPD of a hierarchical variable **H** conditioned on its parents, we assume that each class in the hierarchy of **H** has a probability distribution over its immediate subclasses that is conditioned on (some of) the parents of **H**. We can treat each of these classes as a simple variable. Thus, the problem of representing a (conditional) probability distribution over a hierarchical variable reduces to the problem of representing a (conditional) probability distribution over simple variables.

Let F be a simple variable that has both simple and hierarchical parents. Suppose $pa^s(F)$ denotes the simple parents of F and suppose $pa^h(F)$ denotes the hierarchical parents of F. Let $\mathbf{X}$ denote an assignment of a class to each hierarchical parent of F; we call $\mathbf{X}$ a *parent context*. $\mathbf{X} = \left(c_x^1, \ldots, c_x^n\right)$, where c_x^i is a class in the tree hierarchy of the i^{th} hierarchical parent of F.

To represent $P\left(F|pa^s(F) \wedge pa^h(F)\right)$ we specify the default probability distribution over F for different combinations of values for simple parents, $pa^s(F)$, and some parent contexts. The given parent contexts (parent contexts for which the default distribution over F has been defined) can be represented in a concept hierarchy C_p of partial ordering $C_p = (Z_p, \leq)$, where Z_p is the set of given parent contexts. A parent context $\mathbf{X} \in Z_p$ is a *super parent context* of parent context $\mathbf{Y}$, written $\mathbf{Y} \leq \mathbf{X}$, if $\forall i,\ c_y^i \leq c_x^i$. The conditional distribution over F for a parent context $\mathbf{X}$ is inherited from the default probability distribution of F for the[3] super parent context $\mathbf{Z}$ of $\mathbf{X}$ in Z_p that is not overridden by a more specific default probability distribution of F for parent context $\mathbf{Y}$ such that $\mathbf{X} \leq \mathbf{Y} < \mathbf{Z}$.

Phase1: Abstract Consider the Bayesian network shown in Figure 2 (a). The hierarchical variable **LOC** represents the location on earth. The simple variable S represents the season. Figure 2 (b) shows the part of the hierarchy of the location. We consider that S has only two values: (northern hemisphere) *summer* (May – October) and *winter* (November – April). The probability distribution of class *animal* of **LT** over its immediate subclasses can be conditioned on some of its parents. Suppose the distribution of class *animal* in the *equatorial* region is independent of season, so for parent context $\mathbf{X} = (equatorial)$ we could have a default distribution such as:

$P_d(reptile|animal \wedge \mathbf{X}) = 0.1$
$P_d(insect|animal \wedge \mathbf{X}) = 0.4$
$P_d(mammal|animal \wedge \mathbf{X}) = 0.2$, etc.

In the polar regions the distribution of class *animal* over its children depends on the season, so for parent context $\mathbf{X} = (N - polar)$ we could have a default distribution such as:

$P_d(reptile|animal \wedge S = summer \wedge \mathbf{X}) = 0.01$
$P_d(mammal|animal \wedge S = summer \wedge \mathbf{X}) = 0.04$
$P_d(insect|animal \wedge S = summer \wedge \mathbf{X}) = 0.7$, etc.

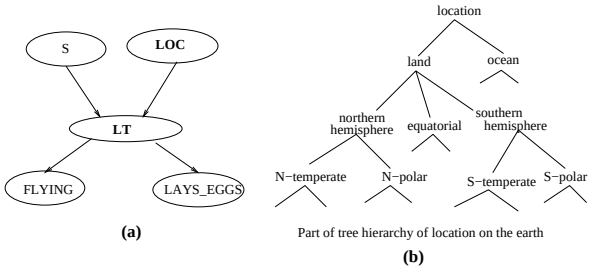

Figure 2: (a) A Bayesian network with simple and hierarchical variables (b) A part of the tree hierarchy for **LOC**

4 Construction of a Flat Bayesian Network

An observation about a hierarchical variable **H** can be *positive* or *negative*. An observation about **H** is positive when we observe that **H** is C_{obs}, where C_{obs} is a class in the hierarchy of **H**. This means that **H** has a value v_h such that $v_h \in C_{obs}$. An observation about **H** is negative when we observe that **H** is $\neg C_{obs}$. This means that **H** has a value v_h such that $v_h \notin C_{obs}$. Without loss of generality, we can assume that there is always one positive observation about **H**, denoted by C_{pos}. For example, suppose we have two positive observations $C1$ and $C2$ about **H**. If classes $C1$ and $C2$ are disjoint then observations about **H** are not consistent. If $C1$ is a subclass of $C2$, $C1 \wedge C2$ equals $C1$. If there are only negative observations about **H**, we assume root class is the positive observation. There can be multiple negative observations about **H** that are descendants of C_{pos} in the hierarchy of **H**.

We assume a query asks either for a distribution over a simple variable or for the probability of a particular class in a hierarchical variable.

Given some evidence (a set of observations) and a query, we construct a flat Bayesian network, B^f, from a hierarchical Bayesian network B^h by replacing each hierarchical variable

[3] We assume that there is only one most-specific super parent context. If there are more than one most-specific compatible parent contexts, with different default probability distributions, there must be a default distribution that disambiguates the two default distributions. We are thus forcing the user to disambiguate cases in which there would otherwise be a problem of multiple inheritance.

with a simple variable whose domain includes only those values necessary to answer the query. The state space of a hierarchical variable **H** can be abstracted because, for a given problem, only certain classes of **H** are supported directly by the evidence or relevant to the query. To construct B^f from B^h given some evidence and a query, we traverse B^h from the leaves upwards, prune those variables irrelevant to answer the query, and abstract the hierarchical variables. We abstract the hierarchical variables as part of a pass to remove variables that are irrelevant to the query. From the bottom-up, you can recursively prune any variable that has no children and is not observed or queried [Geiger *et al.*, 1990; Pearl, 1988] as well as abstract hierarchical variables when their children have been abstracted. We can then answer the query from the flat Bayesian network using any standard probabilistic inference algorithm.

4.1 Abstraction of Hierarchical Variables

Let **H** be a hierarchical variable with values $Val(\mathbf{H})$. The *abstraction* of the domain of **H**, denoted by $part(\mathbf{H})$, is a partition of $Val(\mathbf{H})$. That is, $part(\mathbf{H})$ is a set of subsets of $Val(\mathbf{H})$ such that all sets in $part(\mathbf{H})$ are mutually disjoint and the union of all the sets equals $Val(\mathbf{H})$. We refer to a set in $part(\mathbf{H})$ as an *abstract value* of **H**. An *abstraction of hierarchical variable* **H**, denoted by H^a, is a simple variable with domain $Val(H^a) = part(\mathbf{H})$.

4.2 Finding a Safe Abstraction

In this section we describe a simple algorithm *Flat_Bayes* for constructing a flat Bayesian network B^f from a hierarchical Bayesian network B^h given some evidence and a query. A safe abstraction has the same posterior distribution over the query variable in B^f as in B^h. To develop a safe abstraction for efficient inference, the following constraints are followed in *Flat_Bayes*: 1. For every evidence or query, there must be an abstract value that directly associates with that evidence/query and conveys its effect. 2. All abstract values must be mutually disjoint and exclusive. The algorithm *Flat_Bayes* consists of three phases:

Phase0:
If a query asks for the probability of a particular class in hierarchical variable **H**, we create an extra binary child Q of **H**, which is true exactly when the query is true (i.e., $P_d(Q = true|C_{root}) = 0$ and $P_d(Q = true|C_q) = 1$, where C_{root} is the root class and C_q is the query class in the tree hierarchy of **H**). The variable Q has the same probability as the query class. We thus reduce the problem to one of computing the probability of a simple variable.

Phase1: Abstract
In this phase *Flat_Bayes* traverses B^h from the leaves upwards. It prunes a variable that is not queried or observed and does not have any children [Geiger *et al.*, 1990; Pearl, 1988]. For each unpruned hierarchical variable **H**, the algorithm computes its abstraction H^a as follows:

Case1: H is not observed
The hierarchical variable **H** can be a parent of several simple variables.[4] We need to find the exceptional classes C_k of **H** with respect to each of these children (i.e., classes that are associated with the evidence or the query). The domain of H^a is the set of non-empty abstract values for each exceptional class. The algorithm computes the abstract value, v^a, for every exceptional class C_k as follows:

- $v^a = C_k$, if C_k does not have any exceptional strict subclasses.
- Otherwise, $v^a = C_k - C_1 - \ldots - C_m$, where $C_1, \ldots, C_m$ are the highest exceptional strict subclasses of C_k. This set difference represents the set of all of the values that are in C_k and are not covered by other abstract values.

The abstract values that are empty are removed from the domain of H^a. To recursively abstract the parents of **H**, the algorithm finds those strict super classes of the exceptional classes C_k, which are affected by **H**'s parents.

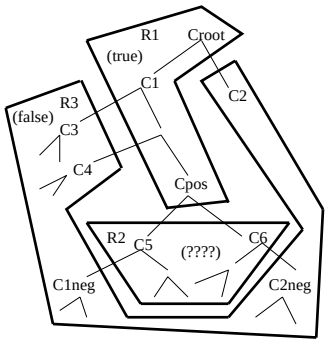

Figure 3: The abstraction hierarchy of **H** is divided into three regions $R1$, $R2$, and $R3$ by positive (C_{pos}) and negative ($C1_{neg}$, $C2_{neg}$) observations about **H**

Case2: H is observed
The observations about **H** divide its tree hierarchy into three regions $R1$, $R2$, and $R3$. $R1$ includes the classes that are true for the observation (superclasses of C_{pos}), $R2$ includes the classes that we know nothing about, and $R3$ includes the classes that we know are false. For example, suppose we have one positive (C_{pos}) and two negative ($C1_{neg}$, $C2_{neg}$) observations about **H** as shown in Figure 3. The regions $R1$, $R2$, and $R3$ are shown by the bold lines in Figure 3. We do not need to distinguish between the values of **H** that we know are false. The values of **H** can be partitioned into two disjoint and exclusive subsets: one (v_{false}) corresponds to all the values that are false and another ($v_{notknow}$) corresponds to all the values we know nothing about. Thus,

$$v_{false} = \{(C_{root} - C_{pos}) \cup (\text{negative observations})\}$$
$$v_{notknow} = \{C_{pos} - (\text{negative observations})\}$$
$$Val(H^a) = \{v_{notknow}, v_{false}\}$$

If **H** also has children, we need to partition its abstract value $v_{notknow}$ into subsets based on the evidence from its children. We need to find the exceptional classes C_k of **H** with respect to its children that are in region $R2$ in the hierarchy of **H**. We

[4] Note that because of the bottom-up nature of the algorithm, **H** can only have simple children at this stage of the algorithm.

can compute the abstract value vo^a that corresponds to each exceptional class C_k in the same way as described in Case1, but we need to remove all the false values. Thus,

$vo^a = v^a - \{$negative observations that are subclasses of$C_k\}$

Let $vo^a_1, \ldots, vo^a_k$ be the non-empty abstract values that correspond to exceptional classes C_k. Let $v_r = vo^a_1 \cup \ldots \cup vo^a_k$. Then, $Val(H^a) = \{vo^a_1, \ldots, vo^a_k, v_{notknow} - v_r, v_{false}\}$

To recursively abstract the parents of **H**, the algorithm finds those strict super classes of exceptional classes C_k and C_{pos} that are affected by **H**'s parents.

Phase2: Construct tables

The algorithm constructs the CPT for variables X^a of a flat network. Let $pa^a(X^a)$ denote the abstracted parents of X^a. We compute $P(X^a = v^a | pa^s(X^a) = V_s \wedge pa^a(X^a) = \mathbf{V})$ for each value v^a of X^a and for each assignment V_s of $pa^s(X^a)$ and $\mathbf{V}$ of $pa^a(X^a)$ for the following two cases:

Case1: X^a is not abstracted but it has abstracted parents

As discussed in Section 3.3, $P(X^a = v^a | pa^s(X^a) = V_s \wedge pa^a(X^a) = \mathbf{V})$ is inherited from the default distribution of X^a for the parent context $\mathbf{C}$, $\mathbf{C} \in Z_p$, $\mathbf{V} \leq \mathbf{C}$ and $\nexists \mathbf{Y} \in Z_p$ such that $\mathbf{V} \leq \mathbf{Y} < \mathbf{C}$, where Z_p is the set of parent contexts for which the default distribution of X^a has been defined.

Case2: X^a is an abstracted variable

Let Δ denote the assignment $pa^s(X^a) = V_s \wedge pa^a(X^a) = \mathbf{V}$. As discussed in Phase1, $v^a = C_k - C_1 - \ldots - C_m$. Then,

$$P(X^a = v^a | \Delta) = P(C_k|\Delta) - \ldots - P(C_m|\Delta)$$

As discussed in Section 3.1, to compute $P(C_k|\Delta)$ we need the probability distribution of all the super classes of C_k over their immediate subclasses for parent context $\mathbf{V}$. As discussed in Section 3.3, we can treat each of these classes as a simple variable. Thus, the probability distribution of a class over its immediate subclasses for parent context $\mathbf{V}$ can be computed in the same way as described in Case1.

The evidence for observed hierarchical variable **H** is translated into the corresponding abstract variable H^a of B^f. The evidence E_h for H^a in B^f is the disjunction of all those abstract values of **H** that are not false for the observation.

The domain size of the variables in B^f is independent of the size of the hierarchies; it depends on how many classes in their hierarchy are *exceptional* with respect to their relevant[5] children in B^h. The running time to construct B^f depends on both the depth and number of the exceptional classes. After constructing B^f we can answer the query from B^f using any standard probabilistic inference algorithm, for example, variable elimination algorithm [Zhang and Poole, 1994] or stochastic simulation.

4.3 Correctness of Flat Bayesian Network

Let B^h be a hierarchical Bayesian network that has n discrete random variables, $\mathbf{X} = \{X_1, \ldots, X_n\}$, and we want to answer some probabilistic query, $P(Q|E)$, where $Q, E \subseteq \mathbf{X}$. Q denotes the *query variables*. We observed that E is in the set e, $e \subseteq domain(E)$, $E \in e$ is the evidence. From B^h we can recursively prune any variable that has no children, and is not observed or queried [Geiger *et al.*, 1990; Pearl, 1988].

$$P(Q|E \in e) = \frac{P(Q \wedge E \in e)}{P(E \in e)}$$

Let $X_1, \ldots, X_s$ be the non-query non-observed random variables of B^h. $pa(X_i)$ denotes the parents of variable X_i. The marginal $P(Q \wedge E)$ can be computed as follows:

$$P(Q \wedge \mathbf{E} \in \mathbf{e}) = \sum_{X_s} \ldots \sum_{X_1} \prod_{i=1}^{n} P(X_i | pa(X_i))_{\{E \in e\}}$$

Lemma 4.1 The posterior probability $P(Q|E)$ computed from a Bayesian network after replacing its variables X_k by their abstractions X^a_k is the same as if it is computed without replacing X_k, if the X^a_k are constructed from the partitions $part(X_k)$ that have the following property :

$$\forall X_k,\ \forall Y_j \in children(X_k),\ \forall V \in part(pa(Y_j)),$$
$$\forall v1, v2 \in V,\ P(Y_j | pa(Y_j) = v1) = P(Y_j | pa(Y_j) = v2) \quad (1)$$

Proof Sketch Summing over all the values of X_k is equivalent to summing over the partition of the values of X_k. Thus,

$$\sum_{v \in Val(X_k)} P(X_k = v | pa(X_k)) \times \prod_{l=1}^{j} P(Y_l | pa(Y_l)) \quad (2)$$
$$= \sum_{V \in part(X_k)} \sum_{v \in V} P(X_k = v | pa(X_k)) \times \prod_{l=1}^{j} P(Y_l | pa(Y_l))$$

If (1) is true, we can distribute the product out of the innermost sum from the RHS of the above equality, leaving the term $\sum_{v \in V} P(X_k = v | pa(X_k))$, which is equal to $P(V | pa(X_k))$. Thus (2) is equal to

$$\sum_{V \in part(X_k)} P(X^a_k = V | pa(X_k)) \times \prod_{l=1}^{j} P(Y_l | pa(Y_l))$$

It is straightforward to incorporate evidence E and query Q into the proof.

Lemma 4.2 The algorithm *Flat_Bayes* partitions the values of the hierarchical variables X_k of B^h such that all its children satisfies the Equation (1).

Proof Sketch Let $part(X_k)$ denote the abstraction of the domain of hierarchical variable X_k computed by *Flat_Bayes*. X^a_k denotes the abstraction of X_k. If X_k is observed, *Flat_Bayes* removes a set V, $V \in part(X_k)$, such that all values in V are false. *Flat_Bayes* constructs the conditional probability tables for variables X^a_k of a flat Bayesian network using "inheritance" for the following two cases:

Case1: X^a_k is not abstracted but some of its parents are

As discussed in Phase2, $P(X^a_k = v^a | pa^s(X^a_k) = V_s \wedge pa^a(X^a_k) = \mathbf{V})$, $\mathbf{V} \in part(pa^h(X_k))$, is inherited from the default distribution of X^a_k for the parent context $\mathbf{C} \in Z_p$, $\mathbf{V} \leq \mathbf{C}$ and $\nexists \mathbf{Y} \in Z_p$ such that $\mathbf{V} \leq \mathbf{Y} < \mathbf{C}$. Thus,
$\forall v \in \mathbf{V}$, $P(X^a_k | pa^s(X^a_k) = V_s \wedge pa^a(X^a_k) = v) = P_d(X_k | pa^s(X_k) = V_s \wedge \mathbf{C})$. This implies Equation (1).

Case2: X^a_k is abstracted

As discussed in Phase2, to compute $P(X^a_k = v^a | pa^s(X^a_k) =$

[5] Variables irrelevant to the query are pruned by *Flat_Bayes*.

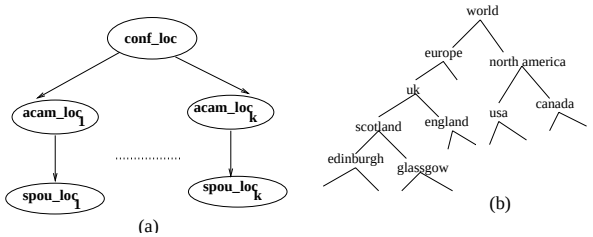

Figure 4: (a) Bayesian network representation of conference academics domain (b) A part of the tree hierarchy for location in the world

$V_s \wedge pa^a(X_k^a) = \mathbf{V}$) we need the conditional distribution of classes C_j, $C_j \in tree(X_k)$, over their immediate subclasses. We can treat a class C_j, $C_j \in tree(X_k)$, as a simple variable. Thus, the proof follows from Case1.

Theorem 4.1 The posterior probability $P(Q|E)$ computed from a flat Bayesian network as constructed by *Flat_Bayes* is equal to the posterior probability computed from a hierarchical Bayesian network.

Proof Sketch The theorem follows from Lemmas 4.1 and 4.2.

5 Experiments

We implement our *Flat_Bayes* algorithm on top of the variable elimination (*VE*) algorithm, we call it *Flat_Bayes* + *VE*. To get an idea of the performance of *Flat_Bayes* + *VE* compared to *VE*, we investigate how the performance of both algorithms vary with the number of hierarchical variables in the Bayesian network, with the depth of the tree hierarchies of the hierarchical variables, and with the depth of the exceptional classes.

To test the algorithms, we consider a simple conference domain. In this domain, the location of an academic is influenced by the location of the conference and the location of the spouse of the academic is influenced by the location of the academic. Let there be k academics. We assume that the locations of the different academics are not independent; they depend on the conference location. We consider that all location variables (conference location or person locations) have the same domain and are hierarchical variables. The Bayesian network representation of this domain is shown in Figure 4 (a). The part of the tree hierarchy for the location is shown in Figure 4 (b). Here we have $2k + 1$ hierarchical random variables. The hierarchical random variable **conf_loc** denotes the conference location. The hierarchical random variables $\mathbf{acam\_loc_1}, \ldots, \mathbf{acam\_loc_k}$ denote the locations of the academics, and $\mathbf{spou\_loc_1}, \ldots, \mathbf{spou\_loc_k}$ denote the locations of their spouses respectively.

To specify the conditional probability $P(\mathbf{acam\_loc_k}|\mathbf{conf\_loc})$, we need to specify the default distribution of each class in the hierarchy of $\mathbf{acam\_loc_k}$ over its immediate subclasses for some parent contexts (for some classes in the hierarchy of **conf_loc**). We assume the conference location influences each academic's location very locally; we need to specify the default distribution of the classes over their immediate subclasses for very few parent contexts.

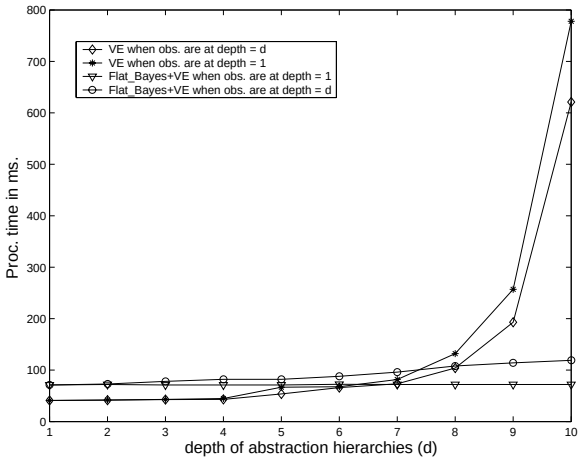

Figure 5: The average inference time of applying *VE* and *Flat_Bayes* + *VE* for k = 2 as a function of the depth of the hierarchy, and depth of the observations

Example: Suppose we know that the academic is in the UK but don't know whether he is in Scotland, Wales or England. Now, knowing that the conference is taking place in the UK but not where in the UK, doesn't provide us any information about the location of the academic in the UK. However, if we know that the conference is taking place somewhere in Scotland, we can state that there is a high probability that the academic is in Scotland. We assume that knowing the conference location in Scotland (e.g., in Edinburgh), doesn't provide us any extra information about where they may be if not in Scotland. Thus, we could have a default probability distribution of the class *uk* in the hierarchy of $\mathbf{acam\_loc_k}$ over its immediate subclasses conditioned on **conf_loc** such as:

$$P_d(scotland|uk \wedge \mathbf{conf\_loc} = scotland) = 0.8$$
$$P_d(scotland|uk \wedge \mathbf{conf\_loc} = england) = 0.3$$
$$P_d(scotland|uk \wedge \mathbf{conf\_loc} = world) = 0.4$$

For the experiments, we observe the locations of the spouses and we want to compute where the conference is taking place. To determine how the inference time depends on the number of hierarchical variables, we vary k from 1 to 8. We represent the abstraction hierarchy of all location variables by a binary tree of depth d where d is a parameter. To determine how inference time depends on the depth of the hierarchy (d) we vary d from 1 to 10 with a step of 1. To determine how the inference time depends on the depth of the exceptional classes, for each value of d we vary the depth of the observation for the observed variables from 1 to d. We compute the posterior distribution of **conf_loc** by applying only *VE* and *Flat_Bayes* + *VE*.

Figure 5 shows the average inference time of both *VE* and *Flat_Bayes* + *VE* for the belief network that has only 2 academics as a function of the depth of the tree, and depth of the observations. Figure 6 shows the average inference time of both *VE* and *Flat_Bayes* + *VE* as a function of the number of academics and the depth of observations; here we consider the depth of the hierarchy is 8. Figure 5 shows that the inference time for *Flat_Bayes* + *VE* doesn't depend on the depth of the

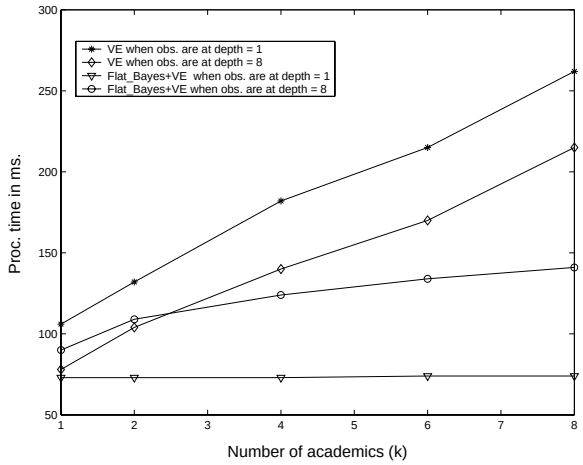

Figure 6: The average inference time of applying *VE* and *Flat_Bayes* + *VE* for $d = 8$ as a function of the number of academics and the depth of the observations

hierarchy but increases linearly as the depth of the observations increases. This increase is because the depth of the observations for $\mathbf{spou\_loc_k}$ variables changes the depth and the number of exceptional classes for $\mathbf{acam\_loc_k}$ and **conf_loc**. Note that *VE* takes less time as the depth of observation increases. This is because the domain of $\mathbf{spou\_loc_k}$ variable reduces as the depth of the observation increases. The results shown in Figures 5 and 6 give us some indication of the overhead of constructing a flat Bayesian network and cases in which it may be more effective to use it rather than VE alone.

6 Related Work

There is very limited work on combining taxonomic hierarchies and probabilistic reasoning. The problem of evidential reasoning in a taxonomic hierarchy of hypotheses was first studied by Gordon and Shortliffe [1985] using the Dempster-Shafer (D-S) theory. Shenoy and Shafer [1986] improved the algorithm proposed by Gordon and Shortliffe and showed that belief functions can be propagated using local computations. Pearl [1986] provides a method for the same problem using Bayesian formalism. In all of these, a very restricted naive Bayes classifier network form was considered. In this paper we concentrate on general Bayesian networks that have both hierarchical and simple variables. Pearl's method involves traversing the whole hierarchy for each observation. We show in this paper that we do not need to do this. In more recent work, Koller and Pfeffer [1998] describe a representation language for combining frame-representation systems and Bayesian networks for representing complex structured domains. They are looking at different aspects of the problem, concentrating on multiple objects, combining first-order logic representation with probabilities.

7 Conclusion

Having the values of a variable hierarchically structured can make reasoning more efficient than reasoning over the set of all values. We show how, given a query and observations, we can dynamically construct a flat Bayesian network from the given hierarchical Bayesian network that can be used in any standard probabilistic inference algorithm. The size of the flat network is independent of the size of the hierarchies; it depends on how many of the classes in the hierarchy are exceptional with respect to children that are observed or have observed descendants in a Bayesian network. Thus it is possible to have hierarchical variables with unbounded or infinite domains. For example, with a spatial hierarchy, as long as we have a way to compute the probability distribution over subclasses, there is no reason not to have a hierarchy that is infinitely detailed. It is only when we ask a query of a class or make observations that we need to restrict the size of the hierarchy.

Acknowledgments

This work was supported by NSERC Discovery Grant OGPOO44121. Thanks to Mark Crowley and Ben D'Andrea for proofreading.

References

[Boutilier *et al.*, 1996] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proceeding of Twelfth Conf. on Uncertainity in Artificial Intelligence (UAI-96)*, pages 115–123, 1996.

[Geiger *et al.*, 1990] D. Geiger, T. Verma, and J. Pearl. d-separation: From theorems to algorithms. In *Proceeding of Fifth Conf. on Uncertainity in Artificial Intelligence (UAI-90)*, pages 139–148, 1990.

[Gordon and Shortliffe, 1985] J. Gordon and E.H. Shortliffe. A method of managing evidential reasoning in a hierarchical hypothesis space. *Artificial Intelligence*, 26:323–57, 1985.

[Koller and Pfeffer, 1998] D. Koller and A. Pfeffer. Probabilistic frame-based systems. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI)*, pages 580–587, 1998.

[Mackworth *et al.*, 1985] A.K. Mackworth, J.A. Mulder, and W.S. Havens. Hierarchical arc consistency: exploiting structured domains in constraint satisfaction problems. In *Computational Intelligence*, volume 1(3), 1985.

[Pearl, 1986] J. Pearl. On evidential reasoning in a hierarchy of hypotheses. *Artificial Intelligence*, 28:9–15, 1986.

[Pearl, 1988] J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., 1988.

[Shenoy and Shafer, 1986] P.P. Shenoy and G. Shafer. Propagating belief functions with local computations. *IEEE Expert*, 1:43–52, 1986.

[Zhang and Poole, 1994] N.L. Zhang and D. Poole. A simple approach to Bayesian network computation. In *Proc. of the 10th Candian Conference on Artificial Intelligence*, 1994.