A Hybrid Bayesian Network Modeling Environment

Thuong Doan[1], Peter Haddawy[1,2], Tien Nguyen[1],
and Deva Seetharam[1]
[1] Decision Systems and Artificial Intelligence Lab (DSAIL)
Department of Electrical Engineering and Computer Science
University of Wisconsin-Milwaukee
Milwaukee, WI 53201, USA
E-Mail: thuong@cs.uwm.edu

[2]Intelligent System Lab (ISL)
Faculty of Science & Technology
Assumption University
Bangkok, Thailand
E-Mail: haddawy@isl.s-t.au.ac.th

Abstract: Bayesian networks are a powerful method for building probability models. But the formalism does not support incremental model development and reuse of models. This is partly due to the fact that Bayesian networks require precise probability values, while incremental model development and model reuse require the ability to abstract probability information. We present a formalism called hybrid Bayesian networks that combines the traditional formalism with that of qualitative probabilistic networks [5]. Qualitative probabilistic networks represent probability information with signs showing directionality of influence between random variables. Our formalism allows a model builder to start by specifying only qualitative influences and then add quantitative information as it is available and as time permits. The modeling environment can infer bounds on unspecified probability values based on those specified and on the type of qualitative influence.
Key words: Bayesian networks, qualitative probabilistic networks, probabilistic reasoning.

1. Introduction

Bayesian networks have become the most popular technique for representing and reasoning with probabilistic information. A Bayesian network is a directed acyclic graph that represents a probability distribution. Nodes represent random variables and arcs represent probabilistic correlation between the variables. The types of paths (and lack thereof) between variables indicates probabilistic independence. Quantitative probability information is specified in the form of conditional probability tables (CPT). For each node the table specifies the probability of each possible state of the node given each possible combination of states of its parents. The tables for root nodes just contain unconditional probabilities.

The key feature of Bayesian networks is the fact that they provide a method for decomposing a probability distribution into a set of local distributions. The independence semantics associated with the network topology specifies how to combine these local distributions to obtain the complete joint probability distribution over all the random variables represented by the nodes in the network. This has three important consequences. First, naively specifying a joint probability distribution with a table requires a number of values exponential in the number of variables. In systems in which interactions among the random variables are sparse, Bayesian networks drastically reduce the number of values required. Second, efficient inference algorithms exist that work by transmitting information between the local distributions rather than working with the full joint distribution. Third, the separation of qualitative representation of the influences between variables from the numeric quantification of the strengths of the influences has a significant advantage for knowledge engineering. In building a Bayesian network model, one can first focus on specifying the qualitative structure of the domain and then focus on quantifying the influences. When finished, one is guaranteed to have a complete specification of the joint probability distribution. Many commercial and free software packages exist for building and performing computations with Bayesian networks. For a list of these see [2].

Rather than build an new network from scratch for each new problem, it is desirable to be able to store and reuse network fragments, just as programmers store and reuse code. But while the structure of a network model that specifies which variables influence one another may be applicable to multiple different situations, the exact numerical probabilities will typically be different in different situations. One option is to store and reuse only the network structure, but we can typically do better than this. Often the "directionality" of influence between variables will also be preserved across various different problems. For example, we may

have a network fragment that models the influence of rain on the water level in a river – the more rain the higher the water level, all other factors being equal. Wellman [5] introduced a representation for modeling such qualitative probabilistic influences, called Qualitative Probabilistic Networks (QPN). In this paper we present an environment that combines QPNs with quantitative Bayesian networks, which we call hybrid Bayesian networks. The idea is to store network fragments with influences represented qualitatively. When building a new model, the network fragments are retrieved and composed together. The model builder can specify some or all of the entries in the conditional probability tables. The model building environment uses the specified qualitative influences and any specified numeric probabilities to infer constraints on the unspecified entries in the conditional probability tables. It also checks the consistency of the specified conditional probability table entries with the direction of qualitative influence. As a first step to build such a system, we present an approach to computing probability bounds and checking consistency based on the information of the influence type of a single connection from a parent node to a child node and the incomplete data of that CPT. The challenge of this problem is how we can exploit as much as possible the incomplete CPT data together with the influence type to specify reasonably tight bounds on probabilities of the missing CPT data.

2. Theoretical background

Definition 1 (Bayesian network) [3]: A Bayesian network is a directed acyclic graph of variables (nodes) with a finite set of mutually exclusive states and connected by directed arcs. To each variable A with some parents $B_1, B_2,\ldots, B_n$, there is attached a conditional probability table $P(A| B_1, B_2,\ldots, B_n)$.

Qualitative probabilistic networks are abstractions of Bayesian networks in which qualitative influences between nodes are represented as positive (+), negative (-), and ambiguous (?). The influences are interpreted in terms of first-order stochastic dominance (FSD). If $F(x)$ and $F'(x)$ denote two cumulative distribution functions, the we say that $F(x)$ *FSD* $F'(x)$ iff $F(x) \le F'(x)$ for all x.

Definition 2 (Qualitative influences) [5]: Let $F(z|x_i,\mathbf{y})$ be the cumulative distribution function of Z given $X= x_i$ and the rest of Z's parent nodes $\mathbf{Y}=\mathbf{y}$. We say that node X *positively influences node* Z, denoted $S^+(X,Z)$, iff $\forall\ x_i, x_j, \mathbf{y}, x_i \le x_j \Rightarrow F(z|x_j,\mathbf{y})$ *FSD* $F(z|x_i,\mathbf{y})$. Node X *negatively influences node* Z, denoted $S^-(X,Z)$, when the direction of stochastic dominance in the definition is reversed. When neither of these relationships holds, we say the influence is ambiguous and denote this as $S^?(X,Z)$.

A *hybrid Bayesian network* is an augmented Bayesian network with information of qualitative influences between nodes and the conditional probability tables of nodes that may contain some precise probability values and some bounds on probability values. As an example, suppose the amount of rain positively influences the water level of a river, denoted as S^+(*Rain*, *Water_Level*). We represent a simple hybrid Bayesian network for this phenomenon in the Figure 1, where the parent node *Rain* positively influences the child node *Water_Level* with the connecting arc marked by the sign +.

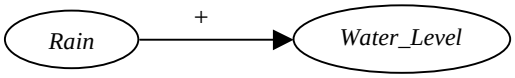

Figure 1: A hybrid BN for the positive influence of *Rain* on *Water_Level*.

The variable *Rain* has four states: *Small*, *Average*, *Heavy*, and *Very_Heavy*; the *Water_Level* variable three states: *Low*, *Medium*, and *High*, both in ascending order. A hypothetical incomplete CPT of the child node *Water_Level* is shown in the Table 1, where the missing data are initially represented as ?.

	Rain			
Water_Level	*Small*	*Average*	*Heavy*	*Very_Heavy*
Low	0.87	?	?	0.02
Medium	?	?	0.43	?
High	0.03	?	?	0.92

Table 1: The partial CPT of *Water_Level* given *Rain*.

3. Algorithm to calculate the probability constraints

This section presents an algorithm to calculate the allowable maximum and minimum probabilities in lieu of missing data in the CPT based on the provided probabilities in that CPT and the type of influence between a parent node A and a child node B. We derive the computations for positive influences, $S^+(A,B)$; the computations for negative influences are similar. Let A and B have discrete states $\{A_1, A_2,\ldots, A_n\}$ and $\{B_1, B_2,\ldots, B_m\}$ in the ascending order, respectively, i.e., $A_1 \le A_2 \le \ldots \le A_n$ and $B_1 \le B_2 \le \ldots \le B_m$. From Definition 2 of qualitative *positive* influence, we have

$P(B \le B_{i*}|A_1) \ge P(B \le B_{i*}|A_2) \ge \ldots \ge P(B \le B_{i*}|A_n)$ **(1a)**

or,

$P(B \ge B_{i*}|A_1) \le P(B \ge B_{i*}|A_2) \le \ldots \le P(B \ge B_{i*}|A_n)$ **(1b)**

for any B_{i*} in $\{B_1, B_2,\ldots, B_m\}$.

By convention, we represent the ascending orders of the parent states $A_1,\ldots, A_n$ in corresponding CPT's columns from left to right and child states $B_1,\ldots, B_m$ in corresponding CPT's rows from top to bottom.

3.1. Calculating minimum values

We divide the minimum constraint computation into three cases: (1) calculating for the cells in the first row, (2) for those in the last row, and (3) for those in middle rows. The 2nd and 3rd have different ways to estimate missing values.

3.1.1. Computing $P(B=B_1|A_{j*})$ for the $j*^{th}$ cell in the first row where $1 \le j* \le n$

In (1a), let B_i* be B_1, the smallest state of B, we obtain the conditions for the first row of the CPT as follows:

$P(B=B_1|A_1) \ge P(B=B_1|A_2) \ge \ldots \ge P(B=B_1|A_n)$ (**2**)

Therefore, in order to obtain the possible minimum value of the missing data at the $j*^{th}$ cell, we simply scan in the first row of the CPT from the $(j* + 1)^{th}$ column to the last column (i.e. from left to right). If we meet any non-missing value, that value is the minimum constraint of $P(B=B_1 \mid A_{j*})$. If we hit the last column and it is still a missing value, the minimum value is 0.

3.1.2. Computing $P(B=B_m|A_{j*})$ for the $j*^{th}$ cell on the last row where $1 \le j* \le n$

In (1b), let B_{i*} sequentially be $B_m, B_{m-1}, \ldots, B_1$, $\forall A_j \le A_{j*}$, we have

$P(B \ge B_m \mid A_j) \le P(B \ge B_m \mid A_{j*})$ (3.m)

…

$P(B \ge B_1 \mid A_j) \le P(B \ge B_1 \mid A_{j*})$ (3.1)

$\Leftrightarrow P(B=B_m|A_{j*}) \ge P(B=B_m|A_j)$ (3.m)'

$P(B=B_m|A_{j*}) + P(B=B_{m-1}|A_{j*}) \ge P(B=B_m|A_j) + P(B=B_{m-1}|A_j)$ (3m-1)'

…

$P(B=B_m|A_{j*}) + \sum_{i=1,m-1} P(B=B_i|A_{j*}) = P(B=B_m|A_j) + \sum_{i=1,m-1} P(B=B_i|A_j)$ (3.1)'

Moving the terms from the left to right hand side, we obtain

$P(B=B_m|A_{j*}) \ge P(B=B_m|A_j)$ (3.m)''

$P(B=B_m|A_{j*}) \ge P(B=B_m|A_j) + P(B=B_{m-1}|A_j) - P(B=B_{m-1}|A_{j*})$ (3.m-1)''

…

$P(B=B_m|A_{j*}) = P(B=B_m|A_j) + \sum_{i=1,m-1} (P(B=B_i|A_j) - P(B=B_i|A_{j*}))$ (3.1)''

Therefore,

$P(B=B_m|A_{j*}) \ge Max \{$

$P(B=B_m|A_j)$, (Term m)

$P(B=B_m|A_j)+P(B=B_{m-1}|A_j)-P(B=B_{m-1}|A_{j*})$,(Term m-1)

…

$P(B=B_m|A_j) + \sum_{i=k,m-1} (P(B=B_i|A_j) - P(B=B_i|A_{j*}))$, (Term k)

…

$P(B=B_m|A_j) + \sum_{i=1,m-1} (P(B=B_i|A_j) - P(B=B_i|A_{j*}))$ (Term 1)

$\} \forall A_j \le A_{j*}$ **(3)**

Let rhs[j] be the right hand side (RHS) of (3), the minimum value of $P(B=B_m|A_{j*})$ is

$\max\{\text{rhs}[j]\}, \ 1 \le j < j*$ **(4)**

Explanation of MinConstraintLastRow: (see Figure 2) We are going to compute $P(B=B_m|A_{j*})$ where $1 \le j<j*$. The minimum value of $P(B=B_m|A_{j*})$ is the maximum of the RHS's of the formula (3). A RHS corresponding to the j^{th} column is represented by rhs[j].

```
MinConstraintLastRow(j*: the column number,
cpt[m][n]: CPT)
  Double rhs[j]: right-hand-side of(3) for j^th
column (1≤ j < j*).
  Double rhs_temp: temporary value of rhs[j]
during computation.
Begin
For each column from the first to the (j*-1)^th do
  (let us call it the j^th column)
  If the cell (m,j) is missing then
    rhs[j] = MinConstraint(j,cpt).
  Else
    Begin
      rhs_temp = cpt[m][j];
      sum = rhs_temp;
      For each row from the (m-1)^th down to 1^st do
      (let us call the i^th row)
        If one of two cells (i,j)&(i,j*) is missing
          then break out of this loop.(*)
        If (cpt[i][j] > cpt[i][j*]) then
            rhs_temp=max{rhs_temp,
                         sum+(cpt[i][j]-cpt[i][j*])}
        sum = sum + (cpt[i][j]-cpt[i][j*]);
      Endfor;
      If rhs_temp is out of the range [0, 1] then
        rhs_temp = 0;
      rhs[j] = rhs_temp;
    End;
Endfor;
Return Max{rhs[j]} with 1 ≤ j ≤ j*-1
End.
```

Figure 2: The algorithm for calculating the minimum constraint of a cell at the $j*^{th}$ column on the last row of a CPT.

During computing, that value is stored temporarily in *rhs_temp*. Let us consider the column j. If $P(B=B_m|A_j)$ is missing in (3), we use the value MinConstraint of that cell as the value of RHS corresponding to the column j. We need to call recursively MinConstraint for that cell. If $P(B=B_m|A_j)$ is not missing, then, since the difference of the term (k) and (k+1) in (3) is equal to $P(B=B_{k+1}|A_j)-P(B=B_{k+1}|A_{j*})$, we repeatedly consider the value of the pairs $P(B=B_i|A_j)$ and $P(B=B_i|A_{j*})$ for every row i. The order of considering is from the row $(m-1)^{th}$ down to the 1^{st}. For a row i, if one of these two values $P(B=B_i|A_j)$ or $P(B=B_i|A_{j*})$ is missing, we stop computing and take the value we get so far (rhs_temp) to assign to rhs[j]. However, since we stop the computation half way, some situations result in a negative value of rhs[j]. Thus we need to take the maximum of that

value and 0. If both $P(B=B_i|A_j)$ and $P(B=B_i|A_{j*})$ exist, we compare them.

If $P(B=B_i|A_j)>P(B=B_i|A_{j*})$, we take the maximum between the maximum value we get so far (rhs_temp) and the current sum. The current sum corresponds to a line (i^{th}) in (3):
$\sum_{t=i-1,m-1}(P(B=B_t|A_j)-P(B=B_t|A_{j*}))+P(B=B_i|A_j)-P(B=B_i|A_{j*})$.
If $P(B=B_i|A_j)\leq P(B=B_i|A_{j*})$, the maximum value that we get so far is still the best. After reaching the 1^{st} row, we have a RHS value of (3) when working with the column j and $j*$. Finally, the possible minimum value of $P(B=B_m|A_{j*})$ is $max\{\text{rhs}[j]\}$ $\forall 1\leq j < j*$.

3.1.3. Computing $P(B=B_i|A_{j*})$ in a middle row i^{th} where $1\leq j*\leq n$, $1<i<m$

In (1b), we repeatedly replace B_{i*} by B_i, B_{i-1}, ..., B_1. Let $S = \sum_{t=i+1,m} (P(B=B_t|A_j)-P(B=B_t|A_{j*}))$, similarly to 3.1.2, we have

$P(B=B_i|A_{j*}) \geq Max \{P(B=B_i|A_j) + S$ (i)
$P(B=B_i|A_j) + S + P(B=B_{i-1}|A_j) - P(B=B_{i-1}|A_{j*})$, (i-1)
…
$P(B=B_i|A_j)+S+\sum_{t=1,i-1}(P(B=B_t|A_j)-P(B=B_t|A_{j*}))$ (1)
} (**5**)

Let rhs[j] be the right hand side of (5),
The minimum value of $P(B=B_i|A_{j*})$ is $max\{$ rhs[j] $\}$ for all $1 \leq j < j*$ (**6**)

<u>*Explanation of MinConstraint:*</u> (see Figure 3) The general idea of this algorithm is mostly the same as 3.1.2, therefore we skip the detailed explanation due to the space limitation of the paper. Computation of the maximum values is similar to computation of the minimum values.

3.2. **Validation and complexity of the algorithm**

Assume that the value $P(B=B_i|A_{j*})$ is missing and its minimum and maximum values calculated by the above algorithms are α and β, respectively. The question is whether any value a such that $\alpha \leq a \leq \beta$ (*) is a valid value for $P(B=B_i|A_{j*})$, i.e. conforms to the definition of its influence type.

For the proof of the minimum computation of positive influence, from (*) and (5), we have

$a = P(B=B_i|A_{j*}) \geq P(B=B_i|A_j) + \sum_{t=i+1,m} (P(B=B_t|A_j) - P(B=B_t|A_{j*}))$
$a = P(B=B_i|A_{j*}) \geq P(B=B_i|A_j) + \sum_{t=i+1,m}(P(B=B_t|A_j) - P(B=B_t|A_{j*})) + P(B=B_{i-1}|A_j) - P(B=B_{i-1}|A_{j*})$
…
$a = P(B=B_i|A_{j*}) \geq P(B=B_i|A_j) + \sum_{t=i+1,m}(P(B=B_t|A_j) - P(B=B_t|A_{j*})) + \sum_{t=1,i-1}(P(B=B_t|A_j) - P(B=B_t|A_{j*}))$
$\Leftrightarrow \sum_{t=i,m} P(B=B_t|A_{j*}) \geq \sum_{t=i,m} P(B=B_t|A_j)$
…
$\sum_{t=1,m} P(B=B_t|A_{j*}) = \sum_{t=1,m} P(B=B_t|A_j)$
$\Leftrightarrow P(B \geq B_i|A_{j*}) \geq P(B \geq B_i|A_j)$
…

$P(B \geq B_1|A_{j*}) = P(B \geq B_1|A_j)$ $\forall j$, $1 \leq j < j*$ (**7**)

That means $a=P(B=B_i|A_{j*})$ satisfies the positive influence property in formula **(1b)**. For the proof of the maximum computation, analogously, we obtain
$P(B \geq B_i|A_{j*}) \leq P(B \geq B_i|A_{j'})$
…
$P(B \geq B_1|A_{j*}) = P(B \geq B_1|A_{j'})$ $\forall j'$, $1 \leq j* < j'$ (**8**)

MinConstraint(i: the row number, j*: the column number, cpt[m][n]: CPT)
Double rhs[j]: right-hand-side of(5) for j^{th} column.
Double S, rhs_temp: temporary value of rhs[j] during computation.
Begin
For each column j from the first to the $(j*-1)^{th}$ **do**
  S = 0;
  For each row t from the $(i+1)^{th}$ to the m^{th} row **do**
    If the cell (t,j) is missing **then**
      break out of this loop.
    Else
      If the cell(t,j*) is missing **then**
        S = S + cpt[t][j]- MinConstraint(t,j*,cpt);
      Else S = S + cpt[t][j] - cpt[t][j*];
  Endfor;
  If there exists any missing cell (t,j) in the above loop **then** rhs[j] = 0
  Else
    Begin
      rhs_temp = cpt[i][j] + S;
      accumulated_sum = rhs_temp;
      For each row t from the $(i-1)^{th}$ down to the 1^{st} **do**
        If one or two cells (t,j),(t,j*) missing **then**
          break out of this loop.(*)
        If (cpt[t][j] > cpt[t][j*]) **then**
          rhs_temp = max{rhs_temp,
            accumulated_sum+(cpt[t][j]-cpt[t][j*])}
        accumulated_sum = accumulated_sum +
                (cpt[t][j]-cpt[t][j*]);
      Endfor;
      If rhs_temp is outranged (0-1) **then**
        rhs_temp = 0;
    End
Endfor;
rhs_temp = max{ rhs[j] } $1 \leq j \leq j*-1$;
sum = $\sum_{(t=1,m \text{ and } t<>i)}$ cpt[t][j*];
If the cell(t,j*) is missing and t > i **then**
  replace cpt[t][j*] in the sum with
  MinConstraint(t,j*,cpt). If it is missing but t < i, ignore it.
Return max{1-sum, rhs_temp}
End.

Figure 3: The algorithm for calculating the minimum constraint of a cell at the $j*^{th}$ column on a middle row i^{th} of a CPT.

However, in order to show the existence of $P(B \geq B_i|A_{j*})$ in inequations (7) and (8), we need to check that in each row i, there exist j and j', $1 \leq j \leq j* \leq j' \leq n$, such that $P(B \geq B_i|A_j) \leq P(B \geq B_i|A_{j'})$. This is

called the *consistency* of the influence type *with* the given data of CPT. In our implementation, we include a function to check this consistency when calculating the probability bounds. If the CPT is inconsistent, the program alerts the user and stops the bounds computation.

When working with a CPT (m rows, n columns) and computing the constraints for an arbitrary cell at the position (i,j), we must go through $m/2$ rows and $n/2$ columns on average. Thus, the complexity is $O(m \times n / 4) = O(m \times n)$.

4. Implementation

We have implemented a prototype hybrid Bayesian network modeling environment in Java. As a demonstration of this prototype, let us come back to the example at the end of Section 2 with the data in the Table 1. For missing data, enter "?" and the program displays it as the bound [0.0, 1.0] (see Figure 4).

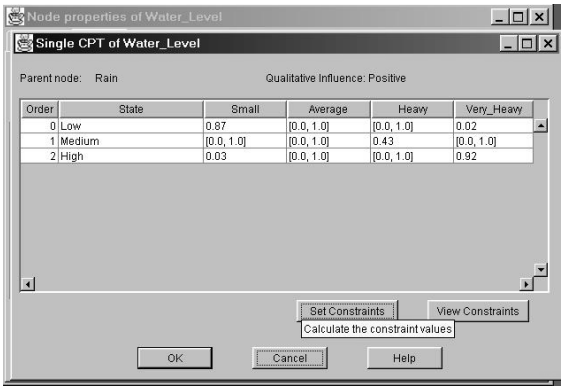

Figure 4. A single CPT of the *Water_Level* node with some missing data.

To calculate the constraints for missing data, we click the button "Set Constraints", then we obtain the resulting CPT in the Figure 5. For example, the probability of *High Water_Level* given *Heavy Rain* is between 0.03 and 0.55. Note that in this example, the program has checked the consistency of positive influence type with the given data and displays the result on the upper-right corner of the window.

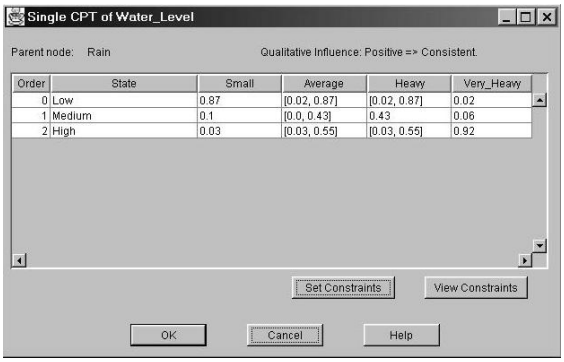

Figure 5. A single CPT of the *Water_Level* node with constraints for missing data.

5. Discussion

The notion of qualitative probabilistic networks was first proposed by Wellman [5] then the algorithm for propagation of qualitative influences in a QBN was studied by Druzdzel & Henrion [1]. Recently Liu & Wellman [4] have formulated an approximation on Bayesian networks to compute bounds of cumulative distribution functions by locally strengthening or weakening selected CDFs. We have built upon this work to support knowledge acquisition in hybrid BNs. As the first step, we have built algorithms and implemented them in a prototype program to calculate the bounds of missing data in a single CPT of a child node with a parent node and its influence type. We are carrying out research for the case of synergies of any subset of parent nodes. This task is rather complicated since we would like to allow the program to calculate the synergistic CPTs of any combination of parent nodes with different synergistic type such as noisy-or, noisy-and, etc..

6. References

[1] Druzdzel, M. J., and Henrion, M.. Efficient Reasoning in Qualitaive Probabilistic Networks, *Proceedings of the Eleventh Conference on Artificial Intelligence*, July 1993.

[2] Haddawy, P.. An Overview of Some Recent Developments in Bayesian Problem Solving Techniques, *AI Magazine*, Special Issue on Uncertainty in AI, Summer 1999.

[3] Jensen, F.V.. *An introduction to Bayesian Networks*, Springer, 1996.

[4] Liu, C.L., and Wellman, M.P.. Using Qualitative Relationships for Bounding Probability Distributions, *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, July 1998.

[5] Wellman, M.P.. Fundamental Concepts of Qualitative Probabilistic Networks, *Artificial Intelligence* 44:257-303, 1990.