Spring 5-2011

Applied Bayesian Networks

Steven David Spansel
University of Southern Mississippi

The University of Southern Mississippi

APPLIED BAYESIAN NETWORKS

by

Steven David Spansel

A Thesis
Submitted to the Graduate School
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Master of Science

Approved:

Dean of the Graduate School

May 2011

ABSTRACT

APPLIED BAYESIAN NETWORKS

by Steven David Spansel

May 2011

A Bayesian Network is a stochastic graphical model that can be used to maintain and propagate conditional probability tables among its nodes. Here, we use a Bayesian Network to model results from a numerical riverine model. We develop an discretization optimization algorithm that improves efficiency and concurrently increases the overall accuracy of the resulting network. We measure accuracy using a new prediction accuracy criteria that includes an *a posteriori* soft correction. Furthermore, we show that this accuracy quickly asymptotes and begins to show diminishing returns on large data sets.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

Table

Chapter 1

INTRODUCTION

In this thesis, we construct a Bayesian Network for a riverine model. In Chapter 2, we review Bayesian Networks. We also discuss the fact that continuous variables must be discretized [6] before they can be input into our Bayesian Network software toolkit. Furthermore, we include a discussion about the algorithms that Bayesian Network software toolkits use to propagate probabilities.

In Chapter 3, we present existing research relevant to our riverine model. We first discuss a soft discretization method suggested by Ebert-Uphoff [4]. We then cover existing research on probability propagation and the junction tree algorithm.

In Chapter 4, we construct our Bayesian Network for the riverine model. First, we describe the river location to be modeled. Next, we describe the Bayesian Network we created to represent this riverine model. We then explain the metric we developed to measure the accuracy of the network. After this, we describe the method we developed to find an accurate and efficient discretization scheme for our Bayesian riverine network. Finally, we show that it is not necessary to use all available model data to train the network – similar accuracy levels can be attained using modest amounts of input data.

Finally, in Chapter 5, we discuss some ways that our Bayesian network model could be improved in the future. First, we develop a heuristic-based iterative learning algorithm that intelligently chooses data cases with which to train the Bayesian network model. We then discuss Ebert-Uphoff's soft discretization scheme [4], which may lead to a more accurate Bayesian network model. Finally, we suggest ways that our bin prediction accuracy method could be improved.

Chapter 2

BAYESIAN NETWORKS

2.1 Definition

A Bayesian Network is a software tool that encapsulates causal relationships for a system with a finite number of variables. Graphically, a Bayesian Network is represented as a Directed Acyclic Graph (DAG). In this DAG, each node represents a random variable with two or more possible values. Each arc represents a causal relationship between the two variables it connects. Relationships are only maintained between *conditionally dependent* variables – an arc only exists for a given relationship if knowledge of that relationship introduces new knowledge about the behavior of the variables involved. For efficiency, an arc is omitted when it does not alter the results, even if a non-zero covariance relationship exists between the nodes connected by the arc.

Consider, as an example, an idealized weather system (depicted in Figure 2.1) where lightning only exists when there is rain, and thunder only exists when there is lightning. Using transitivity, we can conclude that there is no thunder without rain. However, we do not need the information about rain. While rain is a causative factor in lightning, only the presence of lightning is needed to enable a probability that thunder will also occur. Although there is clearly dependence between thunder and rain, there is no *conditional* dependence [15]. Thus, no arc is needed between the *rain* and *thunder* nodes. The arc from rain to lightning and the subsequent arc from lightning to thunder transitively carries that implication implicitly.

Two variables, A and B, are conditionally independent, given C, if any of the equations in Figure 2.2 hold true [23].

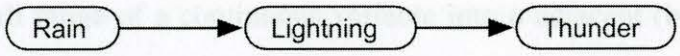


Figure 2.1: Simple weather Bayesian Network. While a non-zero covariance relationship exists between Rain and Thunder, no arc is present, because the same relationship can be derived transitively, passing through the Lightning node.

$$P(A,B|C) = P(A|C)P(B|C)$$
$$P(A|B,C) = P(A|C)$$
$$P(B|A,C) = P(B|C)$$

Figure 2.2: Conditional Probability Equations. Variables A and B are condionally independent if any of these hold true.

In our idealized weather example, the third equation in Figure 2.2 holds true, as shown in Figure 2.3. Thus, we have shown that thunder and rain are *conditionally* independent – knowledge of whether it is raining introduces no new knowledge about the probability of thunder. In these transitive situations, the arc is not necessary and is omitted from the Bayesian network model for efficiency reasons.

In the Bayesian network model depicted in Figure 2.1, the nodes are discrete and binary (i.e., it is either raining or it is not). Generally, a given variable represented by a node in a Bayesian Network may either be discrete or continuous. For the Bayesian Network toolkit that we use, however, all continuous variables must first be discretized. This is done

$$P(Thunder|Rain,Lightning) = P(Thunder|Lightning)$$

Figure 2.3: Conditional Probability Equation, Applied. The probability of Thunder, given the existence of Rain and Lightning, is equal to the probability of Thunder, given only the existence of Lightning. Thus, we learn no new information about the probability of Thunder from the existence of Rain – Thunder and Rain are conditionally independent.

by dividing the full range of a continuous variable into n adjacent (but not necessarilly equally-sized) sub-ranges known as *bins*. The set of bins for a given discretized continuous variable must cover the full range of the continuous variable, with no overlap; they form partitions. This discretization scheme is known as *hard discretization*, and is the scheme used in commonly available Bayesian Network software packages. This is contrasted with *soft discretization* [4], which relaxes the rigidity of the partitions formed by bin boundaries. (For more information, see sections 3.1 and 5.2).

Importantly, once a given variable's values have been discretized, it is no longer possible to determine what the individual value was – that information has been lost. For instance, a value of 3.1 is no longer 3.1. It is simply "something in bin 2, which covers everything from 3 to 5." Thus, once discretized, there is no way to distinguish between a value of 3.1 and a value of 4.9. Loss of accuracy at this discretization step translates to loss of Bayesian network model accuracy. Soft discretization [4] attempts to reduce the impact of this discretization error. In this thesis, we develop additional "soft" methodologies in the same spirit.

2.2 Example: Monty Hall Problem

The Monty Hall problem is a classic probability puzzle. It resembles older probability puzzles such as Bertrand's Box Paradox [2] and the Three Prisoners Problem [7]. The most well-known version of the problem, and its solution, was posed by Marilyn vos Savant – a high IQ columnist – in 1990 [21]. However, her version contained some ambiguities and hidden assumptions. A mathematically explicit version of the problem was posed in 2008 by Krauss & Wang [12]:

> Suppose you're on a game show and you're given the choice of three doors. Behind one door is a car; behind the others, goats. The car and the goats were placed randomly behind the doors before the show. The rules of the game show

> are as follows: After you have chosen a door, the door remains closed for the time being. The game show host, Monty Hall, who knows what is behind the doors, now has to open one of the two remaining doors, and the door he opens must have a goat behind it. If both remaining doors have goats behind them, he chooses one randomly. After Monty Hall opens a door with a goat, he will ask you to decide whether you want to stay with your first choice or to switch to the last remaining door. Imagine that you chose Door 1 and the host opens Door 3, which has a goat. He then asks you "Do you want to switch to Door Number 2?" Is it to your advantage to change your choice?

The intuitive answer to this question is that it does not matter whether the player switches their choice of doors. In fact, after vos Savant published that the correct answer is that it is to the player's advantage to switch, she received thousands of letters in response. Of these letters, 92% of letters from the general public, and 65% of letters from universities, claimed that her solution was wrong [21]. It has since been proven, in many different ways, however, that the correct answer is indeed that it is to the player's advantage to switch. Here, we simultaneously use this puzzle to demonstrate an example Bayesian Network, and use that Bayesian network model to show that the correct answer to the puzzle is that it is to the player's advantage to switch.

This "Monty Hall" network contains 3 nodes: the door initially chosen by the player, the door shown by Monty Hall, and the door containing the prize. Note that the three nodes may not represent three different doors; they may only represent two. This is because the door chosen by the player may be the same door as the one containing the prize; these two decisions are independent of one another. Monty, however, is guaranteed to open neither of these doors. Each node represents a variable that can take on 3 possible values: 1, 2, and 3, representing the 3 doors. Initially (in the absence of any findings), each node has an equal probability distribution. The full Bayesian network model structure is shown in Figure 2.4, and the Conditional Probability Tables are shown in tables 2.1, 2.2, & 2.3.

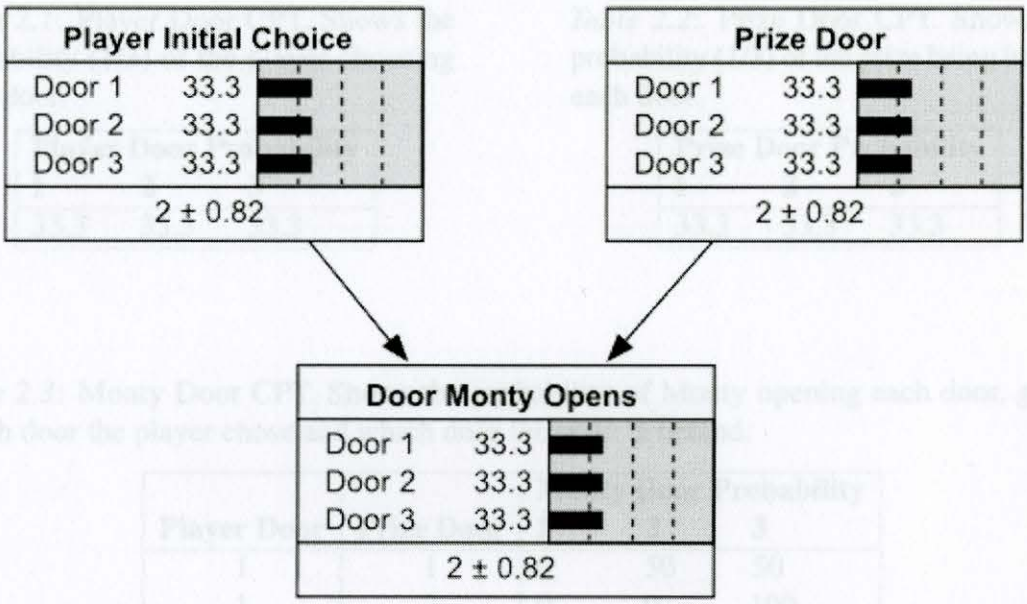


Figure 2.4: "Monty Hall" Bayesian network. In the absence of any findings, the probabilities of all values is 1/3 for all nodes.

When we enter information (findings) about the state of the puzzle, the node probability tables change accordingly. For instance, when the user selects door 1, the probability of Monty opening that door drops to 0%. The probabilities for the prize door node remain unchanged, however, since the player's choice does not affect which door contains the prize. Figure 2.5 depicts a situation in which the user has selected door 1, and the prize is behind door 2. Clearly, Monty will not open the door the user has selected. Since he knows that the prize is behind door 2, he also will not open that door. Thus, there is a 100% chance that Monty will open door 3.

We now know which door the player chose (door 1), and which door Monty Hall opened (door 3), but not which door the prize is behind. When we enter these findings, the network propagates them and calculates a new probability table for the prize node (Figure 2.6). This new probability table shows that there is a 66.7% chance the prize is behind door 1, and only a 33.3% chance the prize is behind the player's door (door 1). Clearly, it is to the player's advantage to switch.

Table 2.1: Player Door CPT. Shows the probability (1/3) of the player choosing each door.

Player Door Probability		
1	**2**	**3**
33.3	33.3	33.3

Table 2.2: Prize Door CPT. Shows the probability (1/3) of the prize being behind each door.

Prize Door Probability		
1	**2**	**3**
33.3	33.3	33.3

Table 2.3: Monty Door CPT. Shows the probability of Monty opening each door, given which door the player chose and which door the prize is behind.

		Monty Door Probability		
Player Door	**Prize Door**	**1**	**2**	**3**
1	1	0	50	50
1	2	0	0	100
1	3	0	100	0
2	1	0	0	100
2	2	50	0	50
2	3	100	0	0
3	1	0	100	0
3	2	100	0	0
3	3	50	50	0

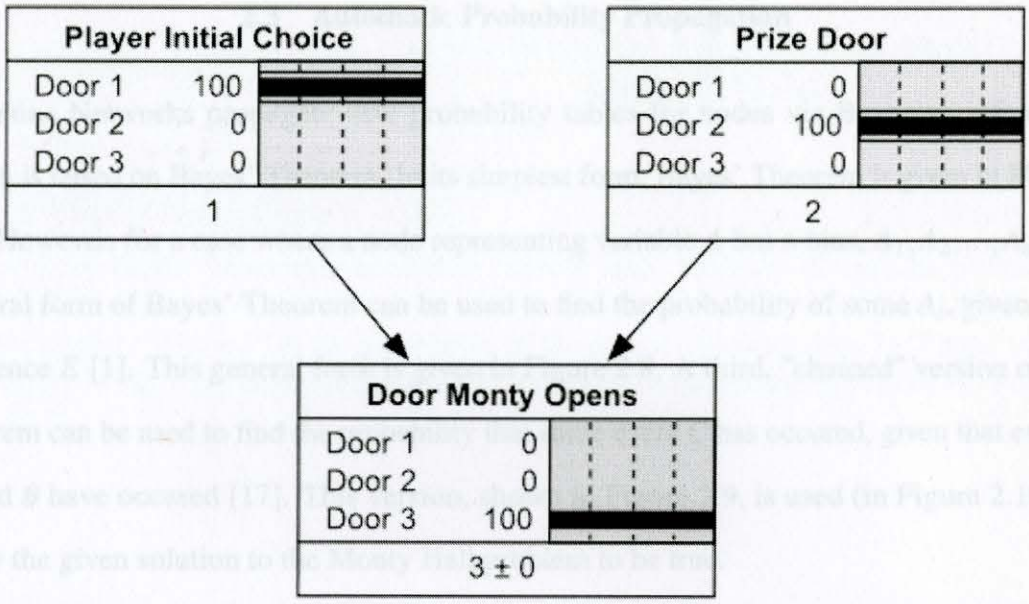


Figure 2.5: "Monty Hall" Bayesian Network. Same as Figure 2.4, but the player has chosen door 1, and the prize is behind door 2.

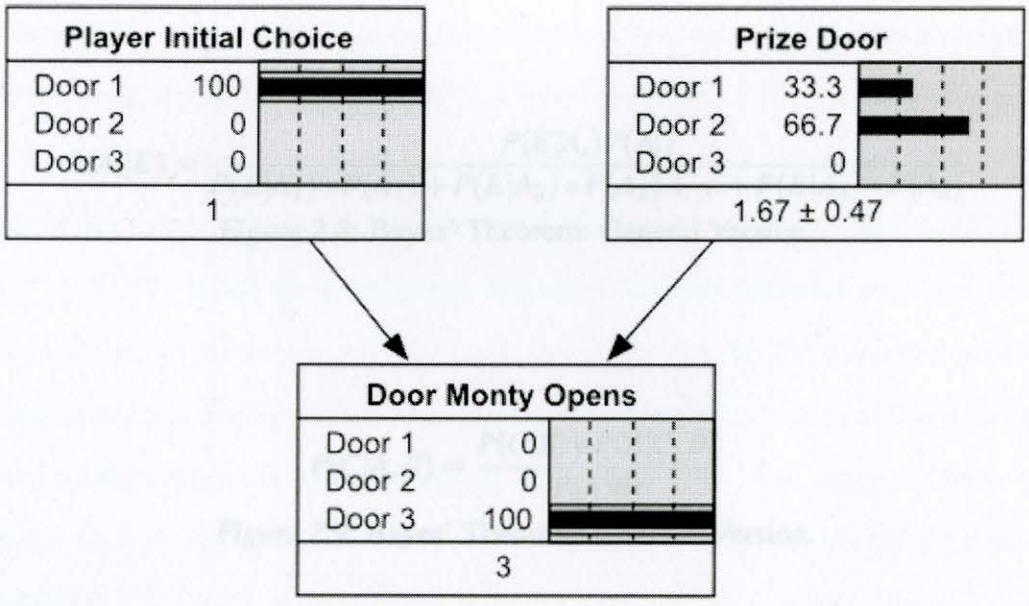


Figure 2.6: "Monty Hall" Bayesian Network. Same as Figure 2.4, but the player has chosen door 1, and Monty has opened door 3. Shows the calculated probability of the prize being behind each of the three doors.

2.3 Automatic Probability Propagation

Bayesian Networks propagate new probability tables for nodes via Bayesian inference, which is based on Bayes' Theorem. In its simplest form, Bayes' Theorem is given in Figure 2.7. However, for a case where a node representing variable A has n bins, $A_1, A_2, ..., A_n$, the general form of Bayes' Theorem can be used to find the probability of some A_i, given new evidence E [1]. This general form is given in Figure 2.8. A third, "chained" version of the theorem can be used to find the probability that some event C has occured, given that events A and B have occured [17]. This version, shown in Figure 2.9, is used (in Figure 2.10) to show the given solution to the Monty Hall problem to be true.

$$P(Forecast|Observations) = \frac{P(Observations|Forecast) * P(Forecast)}{P(Observations)}$$

Figure 2.7: Bayes' Theorem: Simple Version.

$$P(A_i|E) = \frac{P(E|A_i)P(A_i)}{P(E|A_1) * P(A_1) + P(E|A_2) * P(A_2) + ... + P(E|A_n) * P(A_n)}$$

Figure 2.8: Bayes' Theorem: General Version.

$$P(C|A,B) = \frac{P(C|B) * P(A|C,B)}{P(A|B)}$$

Figure 2.9: Bayes' Theorem: Chained Version.

$$P(\text{Prize2}|\text{Monty3},\text{Player1}) = \frac{P(\text{Prize2}|\text{Player1}) * P(\text{Monty3}|\text{Prize2},\text{Player1})}{P(\text{Monty3}|\text{Player1})}$$

$$P(\text{Prize2}|\text{Monty3},\text{Player1}) = \frac{0.33 * 1.0}{0.5}$$

$$P(\text{Prize2}|\text{Monty3},\text{Player1}) = 0.67$$

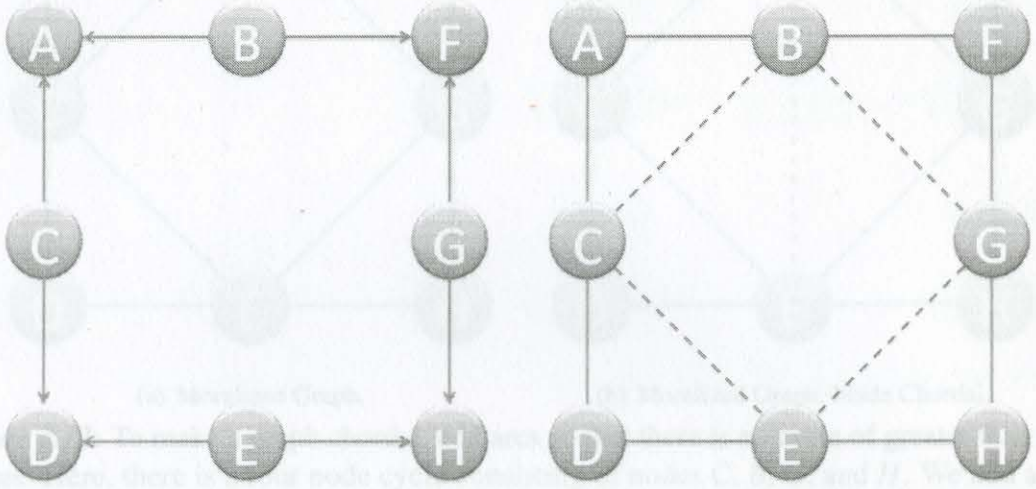
Figure 2.10: Bayes' Theorem, Applied to "Monty Hall" Problem. Shows the probability of the prize being behind door 2, given that the player has chosen door 1 and Monty has opened door 3. "Player1" is the probability of the player choosing door 1. "Prize2" is the probability of the prize being behind door 2. "Monty3" is the probability of Monty opening door 3.

Bayesian inference is the use of Bayes' Theorem to infer knowledge about the state of nodes for which we have no direct knowledge, by propagating knowledge about other nodes in the network. In the Monty Hall example, for instance, the Network inferred, based on the knowledge that the player had chosen door 1 and Monty had opened door 3, that there was a 66.7% chance that the prize was behind door 2.

When findings are entered into the Bayesian Network, the findings must be propagated across the network. For one simple query, as shown in Figure 2.10, this can easily be done using Bayes' Theorem. In a multi-connected network, though, $O(n)$ calculations would need to be performed, and each costs $O(n)$, bringing the total computation complexity cost to $O(n^2)$ [20]. To speed up this process, Bayesian Network software packages use join tree algorithms [13] to decompose the graph represented by the Bayesian Network into a join tree. After this, a propagation algorithm is executed on the join tree. There are many established algorithms [16] for performing this propagation. The specific details of the algorithm used by any specific – excluding open source – implementation is proprietary information.

The first step in the join tree algorithm is to *moralize* the graph [14], shown in Figure 2.11. First, arcs are added between any nodes with a common child (for instance, node A is a common child of nodes B and C). Then, all arrows are changed to nondirectional

connection lines, changing the graph from a directed graph into a more general undirected graph. It is called *moralizing* the graph because parents of nodes are joined together.



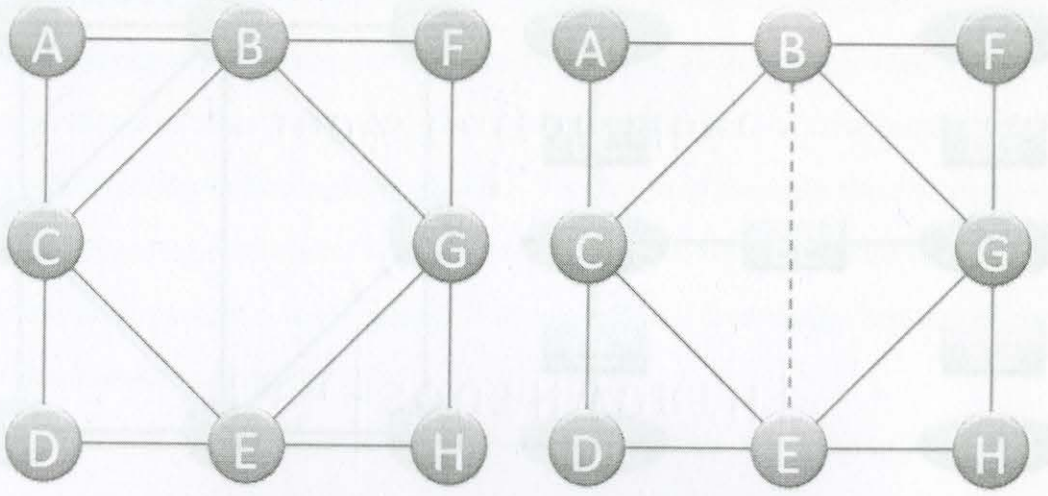
(a) Initial Graph.

(b) Initial Graph, Moralized (Including Direction Removal).

Figure 2.11: To moralize a graph: First, connect any pairs of nodes with a common child (for example, A is a common child of B and C). Then, change all arrows into nondirectional connection lines.

After the graph is moralized, it must be made *chordal* as shown in Figure 2.12. In this step, each cycle consisting of at least four nodes must have *chords* – arcs – added so that there is no cycle of greater than three nodes. In Figure 2.12, there is a four node cycle consisting of nodes C, B, G, and H. We add an arc from B to E, changing the four node cycle into two three node cycles (C, B, E and B, G, E). This is also known as *triangulating* the graph.

Once the graph has been *moralized* and made *chordal*, it is then decomposed into a join tree, also sometimes known as *junction trees* or *clique trees*. A join tree is a representation of the original graph with the original graph's nodes grouped into clusters, each of which represents some subtree of the original graph. The resultant join tree must exhibit the *junction tree property* , which is stated [10] as:



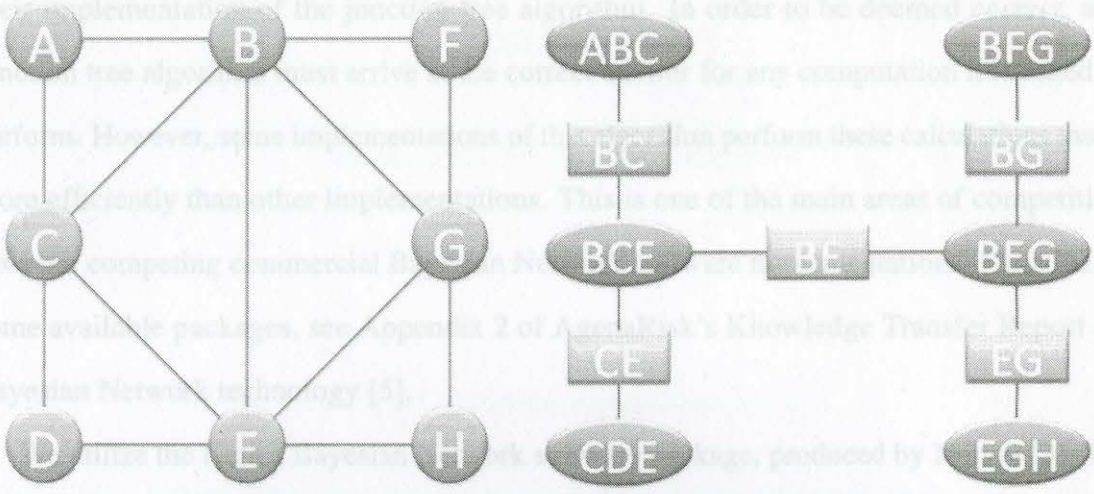
(a) Moralized Graph. (b) Moralized Graph, Made Chordal.

Figure 2.12: To make a graph chordal, add arcs so that there is no cycle of greater than three nodes. Here, there is a four node cycle consisting of nodes C, B, G, and H. We add an arc from B to E, changing the four node cycle into two three node cycles (C, B, E and B, G, E).

> For each pair U, V of [clusters] with intersection S, all [clusters] on the path between U and V contain S.

There are many ways to decompose a graph into a join tree – the results are not unique. It is not hard to find a satisfactory join tree, but it is NP-Complete [10] to find an *optimal* join tree [10]. An example of a moralized, chordal graph decomposed into a join tree is given in Figure 2.13.

After the graph is decomposed into a join tree, a belief propagation algorithm is run on the join tree. There are many different belief propagation algorithms [16]. The specific algorithm used by a particular Bayesian Network implementation to perform this belief propagation is proprietary, and is what separates the fast implementations from the slow implementations. Regardless of the implementation, most Bayesian Network software packages take advantage of the speed increases from decomposing the graph into a junction tree.



(a) Chordal Graph. (b) Chordal Graph, Decomposed into a Join Tree.

Figure 2.13: Decomposing a Chordal Graph into a Join Tree.

2.4 Implementation

Researchers do not need to create Bayesian Network software toolkits in order to be able to use Bayesian network models. Many general use, pre-built implementations are available from a variety of sources. There are many software development firms that produce high quality implementations. These are commercial products, but most also have a free (but limited) version. In addition, several university computer science / mathematics departments maintain free (with license limitations), open-source implementations.

There are two primary characteristics that separate these implementations. The first is the method of use. Many implementations are only available for certain operating systems (i.e. Microsoft Windows). Some implementations can only be used via a graphical user interface (GUI). Such implementations cannot be automated by incorporating them into a computer program. Still other implementations are not wide enough in their scope. For instance, there are some that focus exclusively on using Bayesian Networks to perform sensitivity analysis. The second distinguishing characteristic among Bayesian Network software packages is

their implementation of the junction tree algorithm. In order to be deemed *correct*, any junction tree algorithm must arrive at the correct answer for any computation it is asked to perform. However, some implementations of this algorithm perform these calculations much more efficiently than other implementations. This is one of the main areas of competition between competing commercial Bayesian Network software implementations. For a list of some available packages, see Appendix 2 of AgenaRisk's Knowledge Transfer Report on Bayesian Network technology [5].

We utilize the Netica Bayesian Network software package, produced by Norsys [3]. We use this package for several reasons. First, it is available for all major platforms (Microsoft Windows, Macintosh OS X, Linux, and Sun Solaris). Also, it can be utilized both via a GUI and via Application Programming Interfaces (APIs) available for many different programming languages, including Java, C, C++, .net, and others. Furthermore, it has earned a reputation as being a fast and stable implementation. It has earned this reputation through its performance in other scientific research projects [8] [18] [19]. Finally, we use it because other researchers involved in riverine and oceanographic research projects also use this product [8] [18] [19], allowing easy collaboration.

We use Netica in two forms: as a Java API and as a Windows GUI. Versions of this software can be downloaded from http://www.norsys.com/netica-j.html and http://www.norsys.com/download.html , respectively. Once downloaded, no installation of the software is necessary. The software is downloaded as zip file packages, and they only need to be "unzipped" before they can be used. The versions that can be freely downloaded are limited in size – the number of nodes that can be contained by a network is limited. Furthermore, there are legal restrictions on the use of the software. These can be viewed online at http://www.norsys.com/netica-j.html .

Chapter 3

RELEVANT RESEARCH

3.1 Soft Discretization

Commonly available Bayesian Network software packages utilize hard discretization of continuous variables [4]. In *hard discretization*, each data point belongs to one and only one bin. For instance, if bin 2 of variable X covers the range from 1 to 20, then any data point that falls into that range is assigned to that bin, and any information about how near it is to the edge of the bin is lost. Ebert-Uphoff [4] proposes a probability-based *soft discretization* solution to this problem. Soft discretization is a weighted scheme in which a value does not necessarily belong to one and only one bin —- fractional assignments are allowed. For instance, the value 18 might be given a large weight in bin 2, and a smaller weight in bin 3. In this way, information about the value of the data point is retained stochastically.

3.2 Junction Tree Algorithm and Probability Propagation

The use of the junction tree algorithm for efficient probability propagation in multi-connected networks was first described by Lauritzen & Spiegelhalter [13]. There are now, however, many different algorithms for this purpose, as described by Neapolitan [16]. All of these algorithms begin by *moralizing* the network, making it *chordal*, and decomposing it into a *junction tree*. After that, the algorithms vary as to how they actually perform the propagation of probabilities throughout the network. See, for example, Jensen & Jensen [10], Weisstein [22], and McGlohon [14] to gain a better understanding of this process.

Chapter 4

APPLIED BAYESIAN NETWORKS

4.1 Data Location & SOBEK Model

The United States Geological Survey (USGS) maintains data collection stations at multiple sites along the Kootenai River, located in northern Idaho. To train our Bayesian network model, we utilize output from a numerical model constructed to replicate the river flow properties at the "Tribal Hatchery" site, located approximately 150 miles upstream from the mouth of the river.

A typical river model might require several years to construct, and decades to calibrate. For this reason, we utilize a set of results from an existing, validated numerical riverine model of the Tribal Hatchery site. This model data was generated by the SOBEK model suite, produced by Delft Hydraulics [9]. This powerful software tool simulates one dimensional integral processes in a water drainage system – river, canal, etc [11]. The model was configured specifically for the Tribal Hatchery site mentioned above, and was run over a range of expected conditions. This produced a table of 87,840 cases, each consisting of a value for Discharge, Velocity, Width, and Depth. In this thesis, we simulate having set up and run the model ourselves by selecting cases from this pre-run result set. In the "real world", a researcher would set up and run the model for their specific circumstances. In this case, we used this data because it was readily available, saving the long effort of setting up and running the model ourselves.

4.2 Bayesian Network Representation

One of the earliest studies of river dynamics was based on a study of the Po river made by the Italian engineer Guglielmini in the seventeenth century. These early studies used a transport equation with simply related variables, as shown in Figure 4.1. Such simple equations assume idealized flow. For example, it assumes conservation of water. Yet water may be absorbed by dry ground when a river's depth increases and a dry river bank is first wetted.

$$Q = V * A$$

Figure 4.1: Conservation of Water Equation: A is the cross-sectional area (height times width), V is the velocity of the water, and Q is the discharge.

The presence of small inlets or outlets can also affect our assumption of conservation of water, as can evaporation on a warm day. The simplified equation (see Figure 4.1) also assumes a constant velocity in the vertical dimension. However, water rarely flows with constant velocity from surface to the bottom, and in fact many rivers have a bottom boundary layer that may have significant boundary layer dynamics under high velocity conditions. For example, in 1876, James Buchanan Eads began construction of jetties for the South Pass of the Mississippi River to concentrate the flow, increase the velocity, and take advantage of this significantly modified bottom boundary layer to scour out the bottom of the river, successfully opening the South Pass of the Mississippi River to Ship traffic on a consistent basis for the first time.

In addition, the viscosity of the water in a river system can change with addition of material via solution and suspension, as well as changes in temperature. Velocity can vary as a river's hydraulic gradient changes as well as when it encounters nickpoints. In addition, a river's competence to remove material may add suspended particulate matter in the water column, which alters the volume and density of the water as well. Further, river velocity

varies through pools and riffles and between the thalweg and bankwash. Hence, the equation in Figure 4.1 can sometimes be off by an order of magnitude, so that, while the equation captures river transport from a modeling standpoint, it is an insufficiently accurate model for our purposes.

Unfortunately, our current state-of-the-science numerical models are not sufficiently robust to model the full Navier-Stokes equations (conservation of mass, momentum, and energy) in a fixed, temporally invariant channel either. Hence, even more complex River Models that include the dynamics discussed above are currently beyond our deterministic modeling capabilities. Data, however, includes all of the relevant dynamics implicitly. For these reasons we construct our river estimator using a Bayesian network (shown in Figure 4.2) in which all four of our variables (Width, Depth, Discharge, Velocity) are fully connected. This provides for all interactions between the collected data to be represented stochastically rather than with more simple, but deterministic, equations. It is important to note that, though the arcs in the network are shown as unidirectional arrows, the arrows should be thought of as bidirectional. This is because the algorithms used to propagate probabilities through a Bayesian Network perform this propagation in both directions along arcs. So a change in a "child" node will get propagated to a "parent" node in the same way that a change in a "parent" node will get propagated to a "child" node.

There is insufficient understanding, however, about the amount of data required to train a Bayesian network model. For this reason, we used the more dense output from the SOBEK numerical model, which was based on the original data. The learning algorithm for our network utilizes this simulated data.

The SOBEK variables, while discrete, are not discrete enough – they are discretized on such a fine scale that the values must be re-discretized to a coarser granularity. In section 4.4, we develop an algorithm for choosing an accurate discretization scheme for our network. After discretizing the network, we next created Conditional Probability Tables for all nodes. This step, commonly known as *training* the network, was done using Netica's built-in

algorithm for learning Conditional Probability Tables (CPTs). Netica creates these CPTs by taking in an input file of many cases, and computing the CPTs based on the discretization that was previously applied to the network. For this training step, we used the full case file containing all 87,840 cases of SOBEK model output. The resultant CPTs are depicted in Figure 4.2. In this figure, the likelyhood of each variable being in a specific bin (in the absence of any findings) is indicated by the percentage associated with the specified bin.

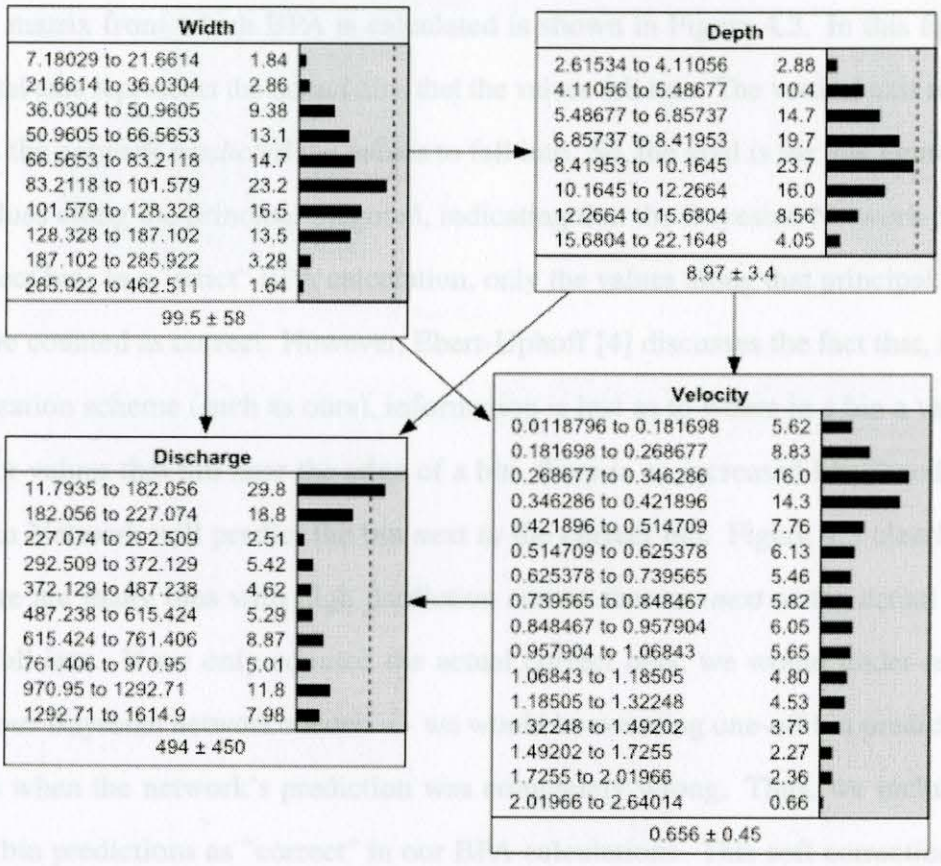


Figure 4.2: Bayesian network representing SOBEK model variables (Width, Depth, Discharge, Velocity). The number of bins and bin boundary placement for each variable was selected using the discretization algorithm described in Section 4.4.

4.3 Performance Metric: Bin Prediction Accuracy

The purpose of the Bayesian Network is to be able to estimate values for all unknown variables, given findings for known variables. To *test* the network, we give the network values for the other 3 variables (but not for the variable being tested) for all 87,840 cases. We then compare the bin which the network estimates the value to be in (for each case) with the bin the value was actually in. We use a modified Bin Prediction Accuracy metric as our performance criteria – the percentage of times the network accurately chooses the correct bin.

The matrix from which BPA is calculated is shown in Figure 4.3. In this figure, the horizontal axis represents the *actual* bins that the values fall into. The vertical axis represents the bins the network *predicted* the values to fall into. So, the goal is for this figure to have high values along the principal diagonal, indicating that the Bayesian Network predicted the correct bin. In a "strict" BPA calculation, only the values along that principal diagonal would be counted as correct. However, Ebert-Uphoff [4] discusses the fact that, in a hard discretization scheme (such as ours), information is lost as to where in a bin a value falls. Thus, for values that fall near the edge of a bin, there is an increased likelihood that the Bayesian Network will predict the bin *next to* the correct bin. Figure 4.3 clearly shows that there are many bins with high prediction counts that are *next to* the actual bins the values fall into. If we only counted the actual correct bins, we would under-report the skill of our Bayesian network model —- we would be counting one-off bin predictions the same as when the network's prediction was completely wrong. Thus, we include those one-off bin predictions as "correct" in our BPA calculations. This soft correction is an *a posteriori* accommodation, in contrast to, but in the same spirit as, Ebert-Uphoff's *a priori* soft discretization [4].

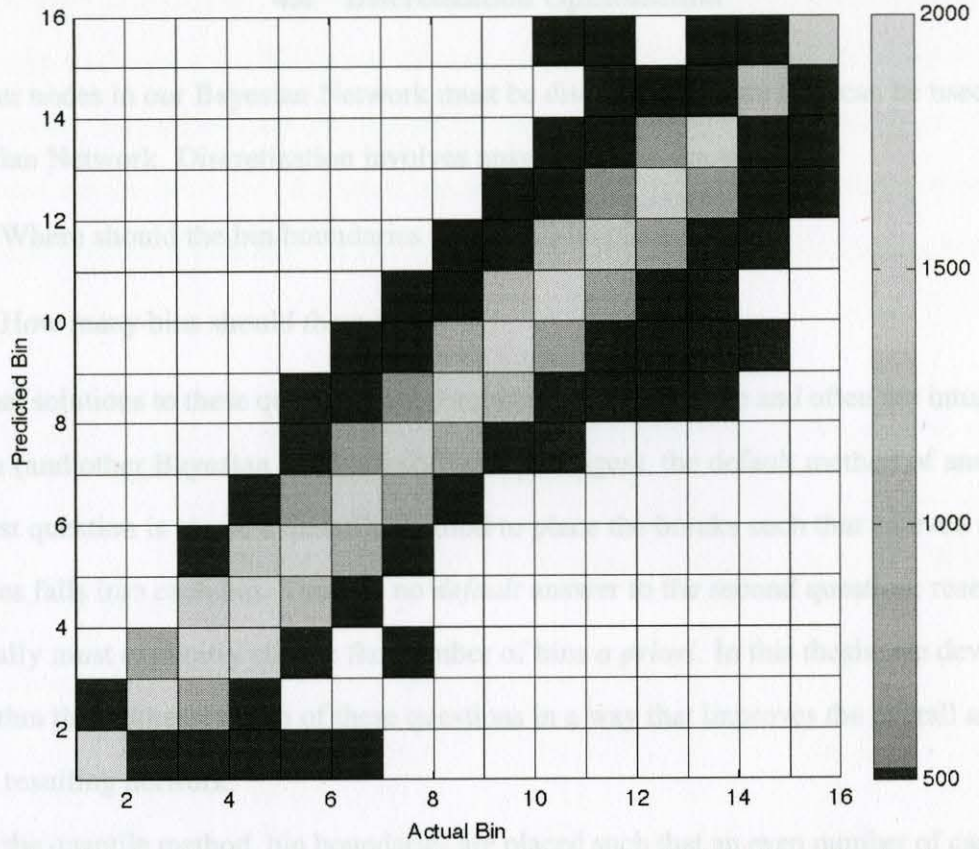


Figure 4.3: Bin Prediction Accuracy: Actual Bin vs Predicted Bin. The horizontal axis represents the *actual* bins that the values fall into, and the vertical axis represents the bins the network *predicted* the values to fall into. There is a strong trend along the principal diagonal.

4.4 Discretization Optimization

All four nodes in our Bayesian Network must be discretized before they can be used by the Bayesian Network. Discretization involves answering two questions:

- Where should the bin boundaries ("breaks") be placed?
- How many bins should there be?

Optimal solutions to these questions are computationally intensive and often not intuitive. In Netica (and other Bayesian Network software packages), the default method of answering the first question is to use a quantile method to place the breaks such that an even number of cases falls into each bin. There is no *default* answer to the second question; researchers generally must explicitly choose the number of bins *a priori*. In this thesis, we develop an algorithm that addresses both of these questions in a way that improves the overall accuracy of the resulting network.

In the quantile method, bin boundaries are placed such that an even number of cases falls into each bin. Essentially, it orders the cases, then splits them into evenly spaced groups. For instance, if we were using 20 bins, the first 5% would go into bin 1, the next 5% into bin 2, etc.

It seems intuitive that placing bins such that an even number of cases fall into each bin would yield high BPA. The problem, however, lies in the fact that the bin boundaries are placed at *pre-determined* places in the Cumulative Distribution Function. So bin boundaries can (and often do) get placed in the middle of a dense cluster of cases that would preferably be grouped together in one bin because they represent similar dynamics.

A data clustering approach addresses this bin boundary placement directly. This method locates these dense clusters of cases, and places the bin boundaries such that the clusters are binned together, forming natural breaks. As evidenced in table 4.1, the optimal bin count combination using this method yields approximately 8% higher Bin Prediction Accuracies than the optimal bin count combination using the quantile method.

For verification testing, we used a brute-force approach to determine the optimal number of bins for each variable. We tested all even numbers between 6 and 20 as bin counts for each variable, using both methods (quantile and data clustering) mentioned above. For each bin count combination, we recorded the BPA for each variable. For each bin boundary placement method, we choose the bin count combination that yielded the highest mean BPA.

Table 4.1: Discretization Optimization Results. Overall, using the optimal number of bins with the cluster method increased accuracy from 90.1% to 98.1%, as compared to using the "default" (20) number of bins with the quantile method.

	Quantile			Cluster		
	Default	Optimal	% Increase	Default	Optimal	% Increase
# of Discharge Bins	20	6	–	20	10	–
# of Velocity Bins	20	8	–	20	16	–
# of Depth Bins	20	6	–	20	8	–
# of Width Bins	20	6	–	20	10	–
Discharge BPA	83.3	96.0	15.3	82.7	99.5	20.4
Velocity BPA	96.8	91.7	-5.3	98.2	96.0	-2.2
Depth BPA	88.3	88.2	-0.1	90.0	97.6	8.4
Width BPA	92.1	92.2	0.1	92.7	99.3	7.1
Mean BPA	**90.1**	**92.0**	**2.1**	**90.9**	**98.1**	**7.9**

Discretization optimization leads to increased prediction accuracy — bins matter. But our discretization approach also yields another benefit. Because it may lead to the use of fewer bins per variable, discretization optimization may yield faster computing times for the resulting Bayesian network model, since there are less bins across which the junction tree algorithm must propagate probabilities.

4.5 Diminishing Returns: Use Less Data

Using all 87,840 cases from the SOBEK model output, the Bayesian Network is able to predict a single unknown variable with 98.1% accuracy, given values for the other three variables, as shown in table 4.1. Here, however, we show that the Bayesian Network can

reach the same prediction accuracy levels, but using far fewer than the entirety of the 87,840 cases. In this case, the network achieved nearly the same BPA using only about 2.3% of the full case set. This is shown in Figure 4.4, which shows that the network's ability to accurately predict which bins a missing variable's values will fall into greatly increases as the network is trained with more data, up to about 500 cases. This accuracy increase begins to slow, though, for cases 500 through 1,000, and begins to asymptote after $O(10^3)$ cases. By randomly selecting an increasing number of cases from the full set, we show that the network's ability to accurately predict which bin a variable's value will fall into quickly reaches nearly the same accuracy levels as using all 87,840 cases, and quickly shows diminishing returns – to add more data would be of decreasing benefit.

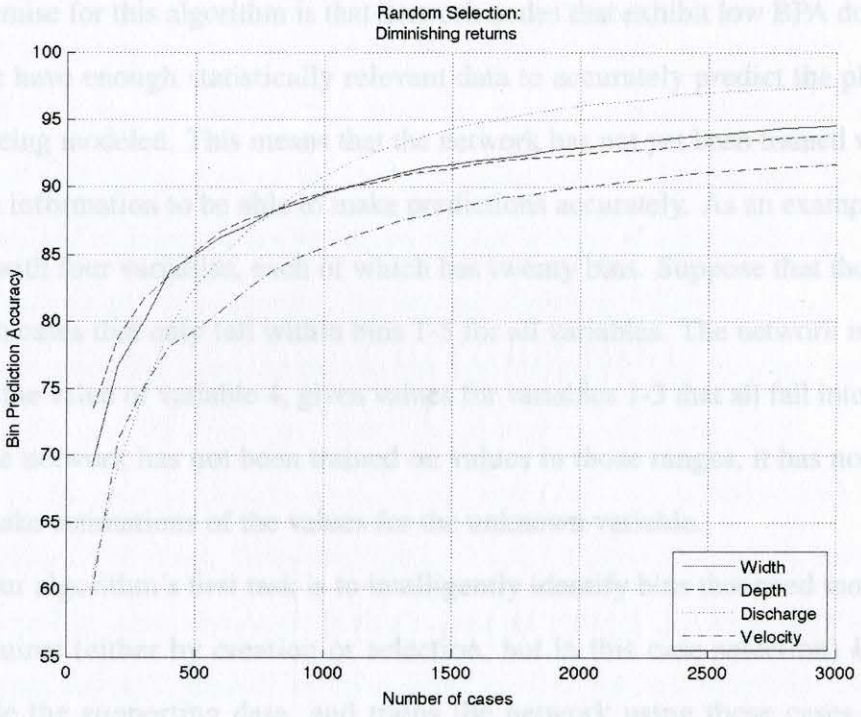


Figure 4.4: Bin Prediction Accuracy vs Number of Cases. The network's ability to accurately predict which bins a missing variable's values will fall into greatly increases as the network is trained with more data, up to about 500 cases. This accuracy increase begins to slow, though, for cases 500 through 1,000, and begins to asymptote after $O(10^3)$ cases.

Chapter 5

FUTURE DEVELOPMENT

5.1 Iterative Learning Algorithm

We showed that the network can achieve essentially the same BPA using far fewer than the full set of 87,840 cases. Thus, there should be an informed way to intelligently choose data points from the full set, outperforming the uninformed random selection method. To illustrate this idea, we built an informed selection algorithm.

The premise for this algorithm is that network nodes that exhibit low BPA do so because they do not have enough statistically relevant data to accurately predict the physical phenomenon being modeled. This means that the network has not yet been trained with enough appropriate information to be able to make predictions accurately. As an example, consider a network with four variables, each of which has twenty bins. Suppose that the network is trained with cases that only fall within bins 1-5 for all variables. The network is then asked to estimate the value of variable 4, given values for variables 1-3 that all fall into bins 15-20. Because the network has not been trained on values in those ranges, it has no basis upon which to make estimations of the values for the unknown variable.

Thus, our algorithm's first task is to intelligently identify bins that need more accuracy. It then acquires (either by creation or selection, but in this case selection) k new cases that provide the supporting data, and trains the network using these cases. It repeats these steps iteratively until the network's bin prediction accuracy levels have risen above a predetermined threshold. In this way, the network experiences informed, rather than random, training, leading to higher BPA using less data. The algorithm is given as follows:

Algorithm 5.1 Iterative Learning Algorithm

Begin

Load net with a *small* initial data set, either randomly selected or representing a known configuration
Bin Prediction Accuracy ← Test Network
while (Bin Prediction Accuracy < Some Predefined Threshold) **do**
Target Bin(s) ← Use heuristic to identify bin(s) to which to add experience
New Cases ← Acquire additional cases that fall into those bins
Train the network to incorporate these cases
Bin Prediction Accuracy ← Test Network
end while

End

Clearly, the Iterative Learning Algorithm depends on having a good heuristic to be able to intelligently make its selections. We developed 2 heuristics, both based upon steepest descent. For both heuristics, we start by testing the network, and recording BPA for all bins of all variables. We then make a copy of the network and *probe* the copy by adding n cases that fit into each bin. Then, we test the copy and record the new BPA, and discard the copy of the network. For the first heuristic, we add m cases to the bin (from the set of bins of *all* nodes) that showed the greatest improvement when probed. For the second heuristic, we add m cases to the bin (from the set of bins of *each* node) that showed the greatest improvement when probed. Results are shown in Figures 5.1 and 5.2. For both heuristics, accuracy increases steeply for the first 500 cases. The accuracy increase slows significantly for cases 500 through 1,000, and asymptotes after case 1,000.

Upon first glance, these results appear to be promising – all variables reach at least 90% accuracy within 2,000 cases. However, Figures 5.3 & 5.4 show that they do not significantly outperform (and in some cases underperform) random selection. In these figures, the dashed line represents the skill attained by using random selection as a heuristic. The solid line represents the accuracy increase (over random selection) attained by the specified heuristic, as a percentage of the skill attained by random selection. In the areas where the solid line is *above* the dashed line, the indicated heuristic is outperforming random selection by the percentage indicated by the solid line. In the areas where the solid line is *below* the dashed

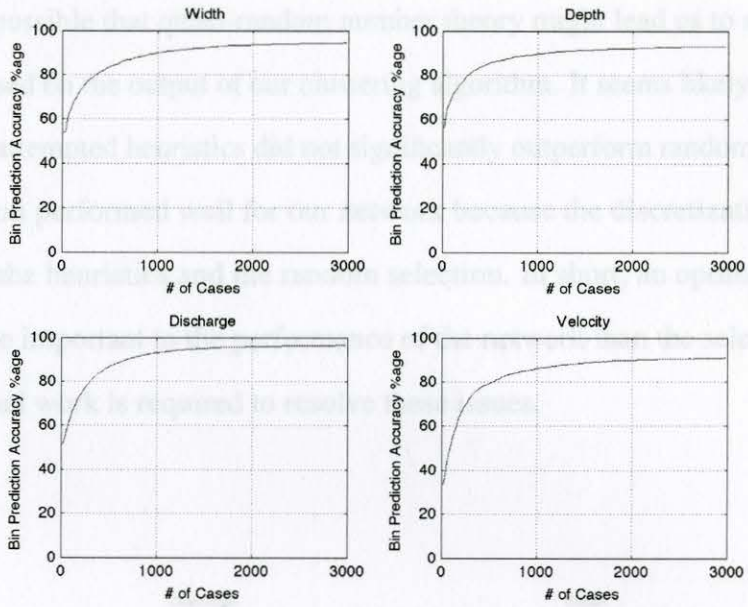


Figure 5.1: Bin Prediction Accuracy: Heuristic 1.

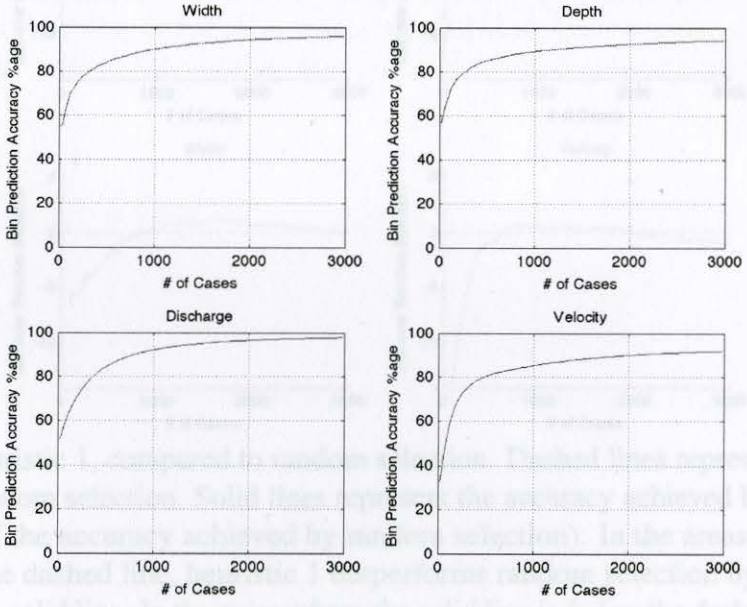


Figure 5.2: Bin Prediction Accuracy: Heuristic 2.

line, the indicated heuristic is underperforming random selection by the percentage indicated by the solid line.

More work needs to be done to find a heuristic that can significantly outperform random

selection. It is possible that quasi-random number theory might lead us to a better heuristic, especially if based on the output of our clustering algorithm. It seems likely that the primary reason that the attempted heuristics did not significantly outperform random selection is that random selection performed well for our network because the discretization optimization benefited both the heuristics and the random selection. In short, an optimal discretization may be far more important to the performance of the network than the selection of training cases. Additional work is required to resolve these issues.

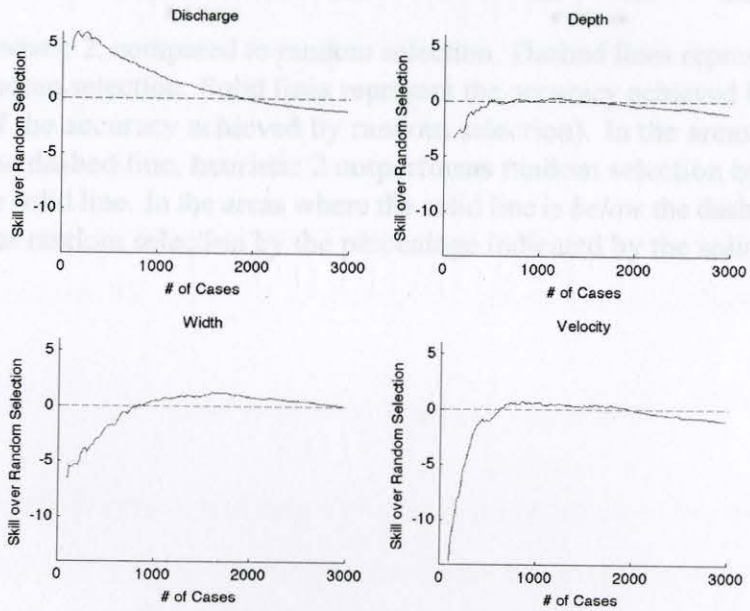


Figure 5.3: Heuristic 1, compared to random selection. Dashed lines represent the accuracy achieved by random selection. Solid lines represent the accuracy achieved by heuristic 1 (as a percentage of the accuracy achieved by random selection). In the areas where the solid line is *above* the dashed line, heuristic 1 outperforms random selection by the percentage indicated by the solid line. In the areas where the solid line is *below* the dashed line, heuristic 1 underperforms random selection by the percentage indicated by the solid line.

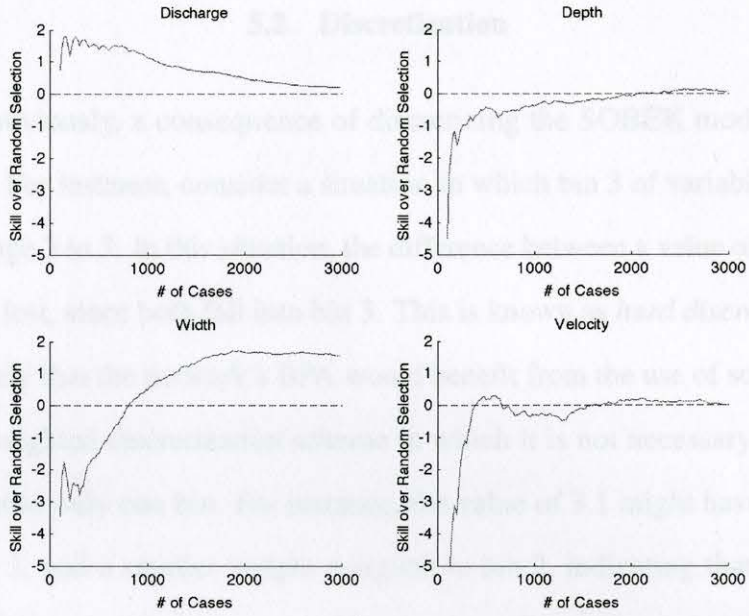


Figure 5.4: Heuristic 2, compared to random selection. Dashed lines represent the accuracy achieved by random selection. Solid lines represent the accuracy achieved by heuristic 2 (as a percentage of the accuracy achieved by random selection). In the areas where the solid line is *above* the dashed line, heuristic 2 outperforms random selection by the percentage indicated by the solid line. In the areas where the solid line is *below* the dashed line, heuristic 2 underperforms random selection by the percentage indicated by the solid line.

5.2 Discretization

As discussed previously, a consequence of discretizing the SOBEK model data is a loss of information. For instance, consider a situation in which bin 3 of variable V contains all values in the range 3 to 5. In this situation, the difference between a value of 3.1 and a value of 4.9 has been lost, since both fall into bin 3. This is known as *hard discretization.*

It seems likely that the network's BPA would benefit from the use of soft discretization [4]. This is a weighted discretization scheme in which it is not necessary to assign all of a value's weight to only one bin. For instance, the value of 3.1 might have a large weight assigned to bin 3, and a smaller weight assigned to bin 2, indicating that it was near the lower bound of bin 3. In this way, we would save some of the information that is now being lost due to the hard discretization scheme. However, as Ebert-Uphoff discusses [4], there are currently no Bayesian Network software packages that allow soft discretization. Thus, a Bayesian Network software package would have to be modified to allow this. This precludes Netica from consideration, since it is a commercial, closed-source application.

5.3 Bin Prediction Accuracy Calculation

As mentioned in the section describing BPA, we include both the bins *next to* and *on* the principal diagonal in Figure 4.3 as correct. It is likely that there is a more optimal way to include these bins in our calculations. Perhaps an exponentially decreasing weight would be more appropriate. In this scheme, the bins along the principal diagonal would be counted as correct with a weight of 1. Bins that are one bin away from the principal diagonal would be counted, but with a smaller weight, and bins that are two bins away would be counted, but with a still smaller weight. More work needs to be done to determine the coefficient weights for an exponentially decaying scheme.

Chapter 6

CONCLUSION

This thesis makes three major contributions to the science of the application of Bayesian Networks. These contributions are the discretization optimization algorithm, the bin prediction accuracy metric, and demonstrating that the accuracy of the network exhibits diminishing returns for large data sets. We used a numerical riverine model to demonstrate the value of these concepts, but the concepts are not limited to this riverine model; they can also be applied to many other numerical models.

Netica provides a default discretization scheme that uses quantiles, usually using the same number of bins for every variable, with an equal number of data samples in each bin. Using this scheme (with twenty bins for each of our four variables), the network achieved 90.1% accuracy. The discretization algorithm introduced here yielded a Bayesian network model that is both more accurate and more computationally efficient. Our algorithm uses data clustering to find natural breaks between groups of clustered data. By changing the discretization scheme and using a cluster selection algorithm to select the number of bins for each variable, we increased the network's accuracy from 90.1% to 98.1%.

We also developed a new accuracy metric for a discretized Bayesian network model. Our new metric is a modified bin prediction accuracy measurement. It is an improvement over a strict bin prediction accuracy metric in that it maintains not only cases in which the network predicted the correct bin, but also cases in which the network was only off by one bin. In a stricter metric, these cases would be counted equally as wrong as cases for which the network predicted a completely wrong bin. This metric is also an improvement over the standard *hindcasting* method of determining a network's accuracy. In hindcasting, the

actual value is compared with the mean of the bin predicted by the network. So, for all cases in which the value does not fall exactly in the center of the bin, the network is penalized as having "error," even though (since it is discretized) it cannot perform any better than predicting the correct bin. In short, hindcasting attempts to extract a continuous value from a discrete variable. Our metric improves on this by staying discrete, avoiding the net residual error incurred by hindcasting.

Finally, we showed that the network's accuracy exhibits diminishing returns as more data is added. This begins a discussion of how to quantify how much data needs to be collected from laboratory and field experiments in order to achieve accuracy that is at least within some pre-defined threshold. Further, the iterative learning algorithm we developed can be used (in conjunction with a numerical model) to determine, *a priori*, approximately how many data points must be collected to achieve a desired accuracy level. This can save researchers much valuable time, money, and resources that could otherwise be wasted on collecting unnecessary data.

It is important to note that the value of these contributions is not limited to the application of Bayesian Networks in a riverine system. Rather, the concepts are general and are applicable to any system in which the data must be discretized before it can be used in a Bayesian network model or similarly discrete tool. Thus, they should be of benefit not only to geoscientists, but also to physicists, engineers, economists, and researchers in any discipline to which numerical modeling can be applied.

BIBLIOGRAPHY

[1] Agena. Bayes Theorem. http://www.agenarisk.com/resources/probability_puzzles/bayes_theorem.shtml .

[2] Edward Barbeau. Fallacies, flaws, and flimflam: The problem of the car and goats. *The College Mathematics Journal*, 24(2):149–154, 1993.

[3] Norsys Software Corporation. Netica Bayesian Network Software from Norsys. http://www.norsys.com/ , 2011.

[4] Imme Ebert-Uphoff. A Probability-Based Approach to Soft Discretization for Bayesian Networks. Technical report, Georgia Institute of Technology, 2009.

[5] Norman Fenton and Martin Neil. Managing Risk in the Modern World: Applications of Bayesian Networks. Technical report, London Mathematical Society and Knowledge Transfer Network for Industrial Mathematics, 2007.

[6] Lawrence D Fu and Ioannis Tsamardinos. A Comparison of Bayesian Network Learning Algorithms from Continuous Data. In *American Medical Informatics Association Annual Symposium*. American Medical Informatics Association, 2005.

[7] Martin Gardner. Mathematical Games column. *Scientific American*, pages 180–182, October 1959.

[8] Cheryl Hapke and Nathaniel Plant. Predicting coastal cliff erosion using a bayesian probabilistic model. *Marine Geology*, 278(1-4):140 – 149, 2010.

[9] Delft Hydraulics. Delft Hydraulics Software: SOBEK Suite. http://delftsoftware.wldelft.nl/index.php?task=blogcategory&Itemid=35 , 2011.

[10] Finn V. Jensen and Frank Jensen. Optimal Junction Trees. In Lopez R. de Mantaras and D. Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 360–366, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[11] Zuwen JI, Huib de Vriend, and Chunhong Hu. Application of SOBEK Model in the Yellow River Estuary. In *International Conference on Estuaries and Coast*, November 2003.

[12] Stefan Krauss and X.T. Wang. The Psychology of the Monty Hall Problem: Discovering Psychological Mechanisms for Solving a Tenacious Brain Teaser. http://www.usd.edu/~xtwang/Papers/MontyHallPaper.pdf , 2003.

[13] Steffen L. Lauritzen and David J. Spiegelhalter. Local Computations with Probabilities on Graphical Structures and their Application to Expert Systems. *Journal of the Royal Statistical Society*, 50(2):157–224, 1988.

[14] Mary McGlohon. Methods and Uses of Graph Demoralization. http://www.cs.cmu.edu/~mmcgloho/pubs/sigbovik3.pdf .

[15] Tom M. Mitchell. Generative and Discriminative Classifiers: Naive Bayes and Logistic Regression. www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf , January 2010. Draft.

[16] Richard E. Neapolitan. Learning Bayesian Networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 5:1–5:1, New York, NY, USA, 2007. ACM.

[17] Daryle Niedermayer. An Introduction to Bayesian Networks and their Contemporary Applications. http://www.niedermayer.ca/papers/bayesian/index.html.old , December 1998.

[18] Nathaniel G. Plant and K. Todd Holland. Prediction and assimilation of surf-zone processes using a bayesian network: Part i: Forward models. *Coastal Engineering*, 58(1):119 – 130, 2011.

[19] Nathaniel G. Plant and K. Todd Holland. Prediction and assimilation of surf-zone processes using a bayesian network: Part ii: Inverse models. *Coastal Engineering*, 58(3):256 – 266, 2011.

[20] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 3 edition, 2009. Available on Google Books.

[21] Marilyn vos Savant. Game Show Problem. http://www.marilynvossavant.com/articles/gameshow.html , 2006.

[22] Eric W. Weisstein. "Chordal Graph." From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/ChordalGraph.html .

[23] Weng-Keen Wong. Bayesian Networks: A Tutorial. http://dimacs.rutgers.edu/Workshops/Surveillance/slides/wong.ppt , 2005.