DECOMPOSING BAYESIAN NETWORK REPRESENTATIONS OF DISTRIBUTED SENSOR INTERPRETATION PROBLEMS USING WEIGHTED AVERAGE CONDITIONAL MUTUAL INFORMATION

by

Benjamin J. Haan

B.S., Illinois State University, 2005

A Thesis
Submitted in Partial Fulfillment of the Requirements for the
Master of Science Degree

Department of Computer Science
in the Graduate School
Southern Illinois University Carbondale
August, 2007

THESIS APPROVAL

DECOMPOSING BAYESIAN NETWORK REPRESENTATIONS OF DISTRIBUTED SENSOR INTERPRETATION PROBLEMS USING WEIGHTED AVERAGE CONDITIONAL MUTUAL INFORMATION

By

Benjamin J. Haan

A Thesis Submitted in Partial

Fulfillment of the Requirements

for the Degree of

Master of Science

in the field of Computer Science

Approved by:

Norman Carver III

Henry Hexmoor

Shahram Rahimi

Graduate School
Southern Illinois University
July 2, 2007

AN ABSTRACT OF THE THESIS OF

Benjamin J. Haan, for the Masters of Science degree in Computer Science, Presented on July 2 2007, at Southern Illinois University at Carbondale.

TITLE: Decomposing Bayesian Network Representations of Distributed Sensor Interpretation Problems Using Weighted Average Conditional Mutual Information

MAJOR PROFESSOR: Dr. Norman F. Carver III

Multi-agent systems (MAS) are systems that consist of many individual agents working together to solve a common goal e.g. large scale problems. *Sensor interpretation* (SI) is a real world problem in which data is collected by a series of sensors distributed throughout a target area. This data is then analyzed to determine which of the known possible events has occurred. SI problems fall in the category of NP, and therefore quickly become intractable as the number of input sensors or events grow. Due to the NP nature of SI problems, solution methods are needed that can either decompose the problem and/or find an approximate solution. One such approach is to decompose the SI problem into smaller sub problems, and distribute them among multiple agents; this is known as *Distributed sensor interpretation (DSI).*

In DSI a set of agents is tasked with processing the sensor data and determining the most likely event that could have occurred. More often then not the agents

are not completely independent of one another. Due to this lack of independence it is often necessary for the agents to communicate by exchanging data or local solutions, to achieve their goal. The goal, and difficult part of designing DSI systems, is decomposing the system in such a way as to create relatively independent subproblems thus reducing communication and computation while keeping the accuracy of interpreting events high.

SI domains are often modeled as *Bayesian networks* (BNs), and the distributed version of these systems can be modeled as d*istributed Bayesian networks* (DBNs). The most studied approach for decomposing BNs into DBNs is the *multiply sectioned Bayesian network* (MSBN) framework. This framework focuses on exact probabilistic inference in DBNs. Approaches such as in Y. Xiang et. al. [6, 7] divide the system using the idea of d-separation. This approach performs well, as the resulting sets are as accurate as the original system. However communication still becomes computationally infeasible as problems grow larger. It is clear that approximation is needed in real world DSI systems.

This thesis investigated the effectiveness/feasibility using the concept of *mutual information* (MI) to determine the appropriate ways to decompose an SI BN. By calculating the MI between subsets of the BNs, we hope to find the subsets that share the least amount of information. This knowledge can then be a guide for decomposing the overall BN into largely independent subproblems. Largely independent means that only a minimal amount of communication is required between the subproblems. With this style of decomposition

we aim to improve over the MSBN approach by adding decomposition flexibility through approximation, while keeping the overall accuracy of the system high. The results are promising, as most decompositions performed with a high degree of accuracy. This process is sill exponential in nature, however, this thesis also describes some methods of approximation that can be added to this technique to improve speed without greatly affecting accuracy.

ACKNOWLEDGEMENTS

Dr. Carver for being my mentor and allowing me to work on truly fascinating research during my time as a graduate student. My mother for supporting me in all my decisions and helping finance my college education. My father for telling me that I was smart enough to do whatever I dreamed. Most of all my loving fiancee, for all her support and hard work proof reading my thesis.

This work is based upon work supported by the National Science Foundation under Grant No IIS-04104945. Any opinions, finding, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ALGORITHMS

List of Formulas

CHAPTER 1

INTRODUCTION

Sensor networks are used for tasks such as collecting weather data, satellite communication, determining network topologies, robot navigation, surveying, and also have many military applications. This growing interest in deploying large-scale sensor networks and the complexity of processing and communicating large amounts of data with limited energy, has lead to a need for better, faster algorithms. Sensor networks focus on processing data to determine what is occurring in the environment. In Artificial Intelligence, this is termed *sensor interpretation* (SI). The *multi-agent systems* (MAS) approach to *distributed sensor interpretation* (DSI) in large sensor networks deals with complex trade offs between solution quality and costs. SI is an intractable problem in large sensor networks, because finding the optimal result, known as *maximum a postoriori interpretation* (MAPI), is an NP-Hard problem.

Often times distributed sensor network problems are modeled as *Distributed Bayesian networks* (DBN). To accomplish the task of finding the MAPI, the agents, each containing a section of the *Bayesian network* (BN), must exchange data or partial solutions to achieve a high-quality globally consistent solution. Communication will typically be

necessary due to interactions among the agents subproblems as well as the agents having direct access to only a subset of the global sensor data. However, communication is slow and/or energy intensive, so it must be minimized.

Decomposition from an SI to a DSI problem is necessary for 4 reasons. First in SI-problems the amount of computation required to interpret events grows exponentially with the number of events that must be modeled. This means that it is important for each agent to restrict the number of possible events it must model to a subset of the global events. Second the more events an agent must model the more data evidence is needed. This means that if an agent does not have the necessary data, communication will be necessary. The result can be a drastic reduction in the speed of the system and the need for high bandwidth network connections. The third reason that subproblems must remain small is, the size of the message grows exponentially so communication can potentially grow exponentially. This means that as the number of events grow, communication between agents also grows exponentially. The fourth reason is parallelism, the ability for separate agents to work on subproblems can greatly decrease the time for a system of agents to arrive at a global interpretation. For these reasons it is important to find an effective way to decompose SI problems into DSI problems.

The most studied approach for decomposing a large BN into a set of distributed BNs is the *multiply sectioned Bayesian network* (MSBN) framework. The MSBN

framework utilizes a concept known as d-separation to create DBNs that can perform the exact same probabilistic inferences as in the original BNs. However, the d-separation requirement severely limits the ability to decompose SI BNs, and will generally still be impractical with large SI BNs. Because of the complexity of SI, it is unlikely that any exact approaches will ever be practical with large-scale problems.

In this thesis, we have investigated approaches for decomposing SI BNs that build upon the basic MSBN framework, but result in approximations of the original BN. This allows for greater flexibility in decomposing SI BNs, which can result in improved time and communication performance, at a slight cost of solution quality. In Cheng et. al. [2] a measure called *mutual information* (MI) was used to determine the structure of *Bayesian networks*. We have adapted the MI for use in determining decompositions of SI BNs where the sub-BNs are largely independent of one another. These sub-BNs can be used to produce an MSBN-like structure (d-separation does not hold), which supports efficient approximate DSI.

We have experimentally evaluated this approach on a large number of randomly generated SI-type BNs. Because of the computational expense of generating and splitting large BNs, we have limited the size of the original BNs to between 2 and 7 events, and limited decompositions into two sub-BNs. Our approach appears to perform well for smaller BNs and shows promise for large networks. The basic approach is exponential in

complexity, however sampling techniques will be discussed that could greatly reduce the time needed to decompose large BNs.

Chapter 2 Is a review of the background knowledge needed to understand the work being done. First is a review of distributed sensor interpretation. This section discusses the difficulties of DSI systems. This section also explains how these systems can be modeled as Bayesian networks.

Next is an explanation of MSBN's and a discussion of concepts such as sepsets, and d-separation. These concepts allow a BN to be broken up into smaller components while still having many of the same properties as the original system. This section will also discuss the downfalls of these types of systems and how they can possibly be improved.

The next section of chapter 2 talks about the idea of *Mutual Information*. This section will define this concept in great detail. We will also cover the concepts of conditional mutual information and weighted average conditional mutual information. This chapter will also discuss how these ideas can be used help decompose a BN.

Chapter 2 then discusses the topics of monotonicity, interpretability, and decomposability. In this section an explanation of how these measurements are useful to BNs is provided. Also in the chapter is a discussion of how these values can be calculated.

Chapter 3 discusses our actual experimentation. In this chapter, our algorithm for decomposing BN will be described, along with the types of files that were generated and

why. Chapter 3 discusses the types of choices that were made, and the types of analysis that were performed. This chapter also covers the system used to determine the accuracy of our experiments.

Chapter 4 is an analysis of the results of our experiments. This chapter describes the different factors that we used to measure our systems performance. This chapter also discusses the effects of monotonicity, decomposability, and interpretability on our system. The chapter will then discuss improvements our system has made over previous systems. Next the chapter will discuss the speed of the algorithm and the problems this poses. The chapter ends with a discussion of sampling techniques we have applied and their results.

Chapter 5 discusses possible improvements to our algorithm. Difficulties with large data sets will be looked at, along with sampling techniques. This chapter mainly focuses on how the speed of our algorithm can be increased.

The conclusion discusses the overall viability of our algorithm. Along with a discussion of how our algorithm is generally useful, but needs improvement.

CHAPTER 2

REVIEW / BACKGROUND

2.1 DISTRIBUTED SENSOR INTERPRETATION

2.1.1 Cooperative Distributed Sensor Interpretation

Cooperative distributed problem solving (CDPS) is a subfield of MAS, where a group of agents work together to divide and solve a given large-scale problem. One set of problems that CDPS is appropriate for are *Distributed Sensor Interpretation* (DSI) problems. In DSI the SI system is decomposed into a set of subproblems, these subproblems are then divided up among a set of agents. The agents then work together gathering and processing sensor data to determine which of a given set of events has occurred in the environment.

DSI has many advantages over a single agent (centralized) system for SI. The first advantage is convenience, as sensors are often scattered over a large area this model tends to be more appropriate. Individual agents can be placed throughout the system to collect data from nearby sensors, reducing the need to transmit data long distances. Another advantage is redundancy. If one of the the agents or network links fail, the other agents can still calculate which events are most likely to have occurred. This leads to another

advantage of DSI, graceful degradation. Because agents can be removed from the system without greatly compromising the results, the system can gracefully degrade if a catastrophic event were to take place. The greatest advantage of DSI systems over single computer systems is time to solution. As SI problems are NP-Hard, decomposing the problem up into smaller subproblems can have a dramatic effect increasing the speed at which a solution can be obtained.

However, for a DSI approach to work, the problem must first be divided up into smaller subproblems. This must be done in such a way as to limit the need for communication between agents as much as possible. Communication cost is the biggest flaw of DSI systems as computation is hundreds of times faster than communication and requires significantly less energy. A successfully divided system will allow each agent to compute its local solution with a minimum amount of communication, without sacrificing the overall quality of the system.

2.1.2 Bayesian Networks

A common representation of SI problems is the *Bayesian Network* (BN). A BN is made up of a series of nodes with causal links between them. Each node in the network represents a random variable and has a conditional probability table based on its causal links. These conditional probabilities determine the likelihood of certain values of the node base on the values of the nodes on the other end of the causal links. Figure 2.1 is an

example of a BN with one of its probability tables shown.

E1	E2	D1
T	T	0.95
T	F	0.93
F	T	0.2
F	F	0.8

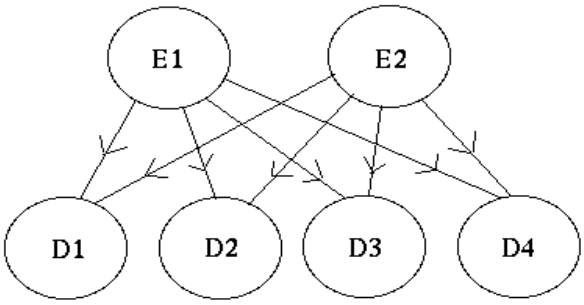

Figure 2.1: Sample Bayesian Network and Probability Table.

In figure 2.1 E1 and E2 represent events in the system such as raining and windy. D1 through D4 represent data from sensors. For example D1 could be a rain sensor, D2 could be a wind sensor and so on. The probability table shown to the left of the BN is for sensor D1, it states, for example, that if E1 is true and E2 is false, then D1 has a 93 percent chance of being true. The table shows the causal relationship between E1, E2 and D1. A similar table exists for D2, D3, and D4. In general, direct links mean direct causation, the standard set up is to have nodes higher up in the network cause nodes lower in the network.

The BN representation may seem simple, however, it can be used to model almost any SI problem. BNs do not require binary events, however, for simplicity sake our representation uses binary valued events. Should a system not consist of binary events, or sensor data, it can still be represented by this type of system. For example, if an event is

either sunny or rainy it could be converted into two binary events: sunny (true and false) and rainy (true and false). In this same manor, continuous sensor can be handled. A sensor that ranges from 10 to 20 degrees with an accuracy of 1 degree could be represented as 10 different sensors ranging from 10 degrees (true or false) to 20 degrees (true or false).

2.2 MULTIPLY SECTIONED BAYESIAN NETWORKS, D-SEPARATION AND SEPSETS

2.2.1 D-Separation

D-separation is a technique that can be applied to BNs using special properties of the network. The d in d-separation stands for dependence, which is what d-separation removes. A set of variables, Z is said to d-separate two groups of variables, X and Y, if the two sets X and Y are independent conditioned on Z. This means that knowledge of X will no longer affect your knowledge of Y once Z is known, thus X and Y are d-separated by Z. D-Separation can occur when all information about X or Y is determined via Z and not from any other variable or path in the network.

1) X -> Z -> Y
2) X <- Z <- Y
3) X <- Z -> Y
4) X -> Z <- Y

Figure 2.2: Possible BN path types.

In figure 2.2 we can see all possible link types between X and Y via Z. The paths in

figure 2.2 do not have to be direct, meaning they can go through other variable along the way, however, they do represent the only path from X to Y via Z. In case 1, X has a causal effect on Z and Z has a causal effect on Y, thus X is indirectly causing Y. This is the same in case 2, however, Y is now indirectly causing X. In these two cases Z d-separates X and Y, as once we know the values of the random variables in Z, X no longer has an effect on Y and vice versa. This is also true in case 3, Z is a common cause of X and Y. For example, if Z was Earthquake, X was John hides under his desk, and Z was Mary hides under her desk. Then, the knowledge of John hiding under his desk should increase the likelihood that Mary will be hiding under her desk. However, once it is known that there was an earthquake, the knowledge of John hiding under his desk no longer affects the probability of Mary hiding under her desk.

Case 4 is a bit different because Z is causally affected by both X and Y. Consider the example where X is a dead battery, Y is no gas, and Z is car won't start. Knowing batter charge tells you nothing about the amount of gas in the tank when you know nothing about the ability to start the car. However, if the car won't start and the battery is not dead this does increase the probability that the gas tank is empty. Thus, Z d-separates X and Y until it is instantiated, when the values for Z are known Z connects X and Y and causes them to be dependent.

2.2.2 Sepsets and Junction Trees

A *junction tree* (JT) is a concept from graph theory in which the graph can be broken up into a set subgraphs connected via sepsets, similar to the model seen in figure 2.3. A sepset is a set of variables that lies between two subgraphs in a JT. A JT is valid if it follows two properties, first every node in the original graph must be represented in the JT, and second all subgraphs must be connected by sepsets containing elements common to both subgraphs. The JT model can be used to represent BNs as they are simply directed graphs. In a BN JT the purpose of a sepset is to limit the number of nodes or agents that must communicate. Sepsets have special properties such as allowing for exact models of the original system in the case of d-sepsets. A d-sepset is formed by the variables that d-separate two portions of a JT. Sepsets can be used to divide a BN as shown in figure 2.3. In figure 2.3, BN_0, BN_1, and BN_2 are portions of the BN while the two squares represent the variables that d-separate each portion. For changes in BN_0 to affect BN_1 the probabilities must be propagated via E_0 and E_2. This means that all communication from BN_0 to BN_1 must pass through the E_0, E_2 sepset.

2.2.3 Multiply Sectioned Bayesian Networks

Multiply sectioned Bayesian Networks (MSBNs) are the most studied approach for building and performing inference in distributed BNs. MSBNs are composed of a group of

interrelated Bayesian subnets. Each subnet in the domain represents a single agent's knowledge. These subnets are then organized into a hypertree structure shown in figure 2.3, connected with d-sepsets. The hypertree structure is essentially a JT with BN subnets for nodes. This model allows for distributed computation and communication, while still allowing for exact inference thus duplicating the original BN. Communication in MSBNs is done by sending updated probability vectors or unnormalized potentials via the sepsets. For example if BN_0 was to communicate with BN_1, This is done in three steps. First the sepset E_0, E_2 probabilities (or potential) is saved. Second the new sepset potentials are computed using data from BN_0. Third using the old sepset and new sepset potential the potential of BN_1 is then updated. After this step BN_0 effect on BN_1 is complete and the message has been passed. For more information on this procedure see section 3.8.3 and Xiang et al. [5]

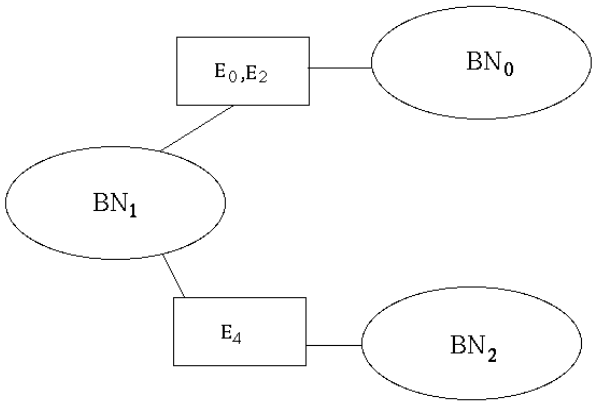

Figure 2.3: An MSBN with corresponding sepsets.

The main problem with traditional MSBNs is the sepsets must d-separate the corresponding subnets. D-separation allows for exact approximation, however, greatly limits the way that BNs can be decomposed. Also D-separation often leads to inefficient systems that have large communication costs. This is due to the communication grown exponentially with the size of the sepset. An example of how d-separation can perform poorly can be seen in section 3.1. It is clear from this that some form of approximation is necessary. Our approach uses similar DBN model, however, focuses on approximation as opposed to exact inference.

2.3 MUTUAL INFORMATION AND VARIATIONS

2.3.1 Mutual Information

In probability theory, *mutual information* (MI) is a measure of the mutual dependence between two sets of random variables. The most common unit for MI is the bit. The MI of two discrete variables X and Y is defined as follows:

$$I(X,Y)=\sum_{y\in Y}\sum_{x\in X}p(x,y)\log_2\left(\frac{p(x,y)}{p(x)p(y)}\right) \tag{2.1}$$

where p(x,y) is the joint probability of x and y occurring and p(x), p(y) are the marginal probabilities of x and y.

In general, MI measures the amount of information shared by X and Y. It is a measure of how much information about one of these variables reduces uncertainty in the

other. This means that if X and Y are independent of one another, then knowing X should tell you nothing about Y. Thus in this case the mutual information between X and Y would be 0. Conversely, if X and Y were the same variable, then knowing X would reveal all information about Y. In this case the MI is the total about of information that can be conveyed in bits. So if X = Y and X is 4bits, then the I(X,Y) is equal to 4bits.

2.3.2 Conditional Mutual Information

MI is a good way of calculating the importance of one variable to another, however, due to the nature of independence relations in BNs and our goal of dividing events this calculation for MI is inappropriate. In our system, the events are independent if no data is known. This means that the MI between event 1 and event 2 will be 0 unless some data is known. This brings us to concept of *conditional mutual information* (CMI). CMI uses conditional probability's in its calculations to add the effect of the conditioning the relationship of the two sets of random variables. CMI is computed as follows.

$$I(X,Y|C)=\sum_{y\in Y}\sum_{x\in X} p(x,y|C)\log_2\left(\frac{p(x,y|C)}{p(x|C)\,p(y|C)}\right) \tag{2.2}$$

where C is a specific instantiation of our set of conditioning variables and p(x,y | C) is the conditional joint probability of x and y, or the probability that x and y occurs given C. p(x | C) is the conditional probability of x occurring given C. This calculation for MI takes into account the affect of the data on the probability of the events. CMI, although better

than MI, does not give us a full picture of the relationship between events in our system. This is due to the fact that it allows for only one possible set of data.

2.3.3 WEIGHTED AVERAGE CONDITIONAL INFORMATION

In our system, there are 2^d possible data combinations, where d is the number of pieces of data, or sensors. Therefor, to calculate the overall effect on one event on another in our system we need a way of computing the CMI over all possible data combinations. To accomplish our goal we created the idea of *weighted average conditional mutual information* (WACMI).

WACMI is a way of computing the average CMI between two events given all possible data combinations, weighted by the probability of each. In DSI, certain data combinations occur more frequently then others. This means that those data combinations are going to be affecting the MI between the two events more often therefore should be weighted as such. WACMI takes the weighting into account and is computed as follows:

$$I(X,Y|C)=\sum_{c\in C} p(c)\left[\sum_{x\in X}\sum_{y\in Y} p(x,y|c)\log_2\left(\frac{p(x,y|c)}{p(x|c)p(y|c)}\right)\right] \qquad (2.3)$$

where p(c) is the probability of current instantiation of the conditioning data, p(x,y | c) is the joint probability of x and y given the current data set, and p(x | c) is the marginal probability of x given the current data set.

Now we have a way to calculate how much of an effect one or more events has on

another in an SI BN. We can use WACMI to break a network into subgroups that have little or no mutual information. If we could find two subgroups of events and data such that WACMI is low, this would mean that each sub group should have little effect on the other. Therefor, should be able to be computed individually with only a small amount of communication. Also, because the groups do not convey a significant amount of information about one another, the decomposed system should be a decent approximation of the original system.

WACMI does have one major drawback, it becomes computationally infeasible as the system grows larger. This is due to the nature of how it is calculated. WACMI must run though all the possible 2^d data sets, where d is the number of data sets. Then for each of these data sets it must also run through 2^e possible event combinations, where e is the number of events. The number of combinations makes the run time for WACMI $O(2^n)$ or exponential, where n is the number of events plus the number of data. This means for WACMI to be applied to larger systems it will have to be approximated. We will discuss approximation via sampling at the end of Chapter 4 and in Chapter 5.

2.4 MONOTONICITY, INTERPRETABILITY, and DECOMPOSABILITY

2.4.1 Interpretability

In a BN we are usually concerned with predicting the most likely set of events given

our data, or MAPI (formula 2.4).

$$MAPI(D) = ARGMAX\left[P(E_i|D)\right] \tag{2.4}$$

However, the most likely set of events may still have only a small probability. For example in a BN with 4 events there are 2^4 possible event combinations, this means the most likely combination can have a probability of just over 1/16. So if the MAPI has a probability of 1/10 this would mean that even if we pick the most likely event we are still incorrect 90 percent of the time, this is unacceptable in real world systems. Real systems must produce accurate results much more then 10 percent of the time. To make sure that we do not have BNs that have low MAPI probabilities we must look at there interpretability

Interpretability is an overall assessment of how well the BN can perform in ideal conditions, given all of the information e.g. how likely are the MAPI. Interpretability is a weighted average of all the possible MAPI given the data (formula 2.5). This is then normalized, i.e., the effect of the all null data set is removed. This is done because in most real systems the data set were nothing is occurring, happens quite frequently and can skew the results.

$$Interp = \sum_{D_i} P(D_i)P(MAPI(D_i)|D_i) \tag{2.5}$$

As the MAPI is the best a system can perform, we will only be looking at systems that have a high interpretability (≥ 90) Interpretability is also the measure by which we

will compare our system, and our other variables such as decomposability and monotonicity.

2.4.2 Event Decomposability

Event decomposability measures how accurately events can be identified when considering the events independently. Event decomposability gives a general idea of how well the overall problem of, identifying the complete set of events that occurred, can be broken down or decomposed into subproblems. These subproblems each then identifying whether individual events occurred. Event decomposability is calculated as follows:

$$EDecomp = \sum_{D_i} \left(P(D_i) \prod_{E_j} \phi(P(E_j|D_i), MAPI(D_i), E_j) \right) \tag{2.6}$$

$\phi(X, Y, Z)$ returns 1 if (X > .5 and Z is a member of Y) or (X < .5 and (not Z) is a member of Y) otherwise it returns 0. Event decomposability is then normalized to remove the effect of the null data set.

If the event decomposability is low, this means that the system performs poorly when events are not considered together. For this reason, we will be using BNs that have high decomposability (over 80 percent). This limits the types of systems for which our technique can be used for, however, it is a realistic measure that must be taken.

2.4.3 Monotonicity

Monotonicity gives a sense of the ability for accurate interpretation of events in a

BN (mean MAPI) with only part of the data. This represents what would happen when a BN is distributed among multiple agents. Domains with a high monotonicity should be easier to decompose, i.e., produce more accurate interpretations with less communication. The monotonicity as defined in Carver & Lesser [1] is calculated as follows:

$$\frac{\sum_{D_i}\left(P(D_i)\sum_{d_j\subseteq D_i}\frac{E(MAPI(d_j),MAPI(D_i))}{2^{|T|}-1}\right)}{\sum_{D_i}P(D_i)} \tag{2.7}$$

Where T is the number of possible data types, D_i ranges over all possible combinations of data, d_j ranges over all non-empty subset of D_i, and E(X,Y) = 1 if x = y else it is 0.

Monotonicity tells us how well the BN preforms in general with only part of the data. BNs that have a low monotonicity will generally require more communication of data when decomposed. The one major drawback of monotonicity is, its calculation has factorial growth rate. The growth rate is due to the fact that it is calculated by testing every possible set of the data. This factorial growth rate means that to calculate monotonicity approximation techniques must be used.

CHAPTER 3

BAYESIAN NETWORK DECOMPOSITION

3.1 PREVIOUS WORK AND SYSTEM GOAL

Successful applications of multi-agent systems to SI usually requires that we be able to decompose the problem into smaller subproblems. These subproblems can then ideally be solved by individual agents (or groups of agents) with as little communication as possible. The purpose is to make systems that are more robust and have a faster time to solution. In previous work done by Y. Xiang et. al. [5] this was done by converting the system into an MSBN using d-separation. This process exploits the structure of the problem to try and gain efficiency when computing exact probabilities. The main problems with Xiang's approach is it only works well on sparse networks. As networks become large and well connected the cost of communication for computing exact probabilities becomes far to great [6].

Consider the fully connected BN with four events in figure 3.1. Suppose we wanted to decompose the network by breaking it in half, so each agent was responsible for interpreting two of the four events. Event dependencies (via the data) would mean that the sepset would have to contain all four events, as would each agent's sub-BN. While the

agents could process their local data in parallel, both the cost to process each piece of data and the cost to communicate probability vectors would be exponential in the number of events in the original BN. Thus, the resulting MSBN would have little or no efficiency improvement.

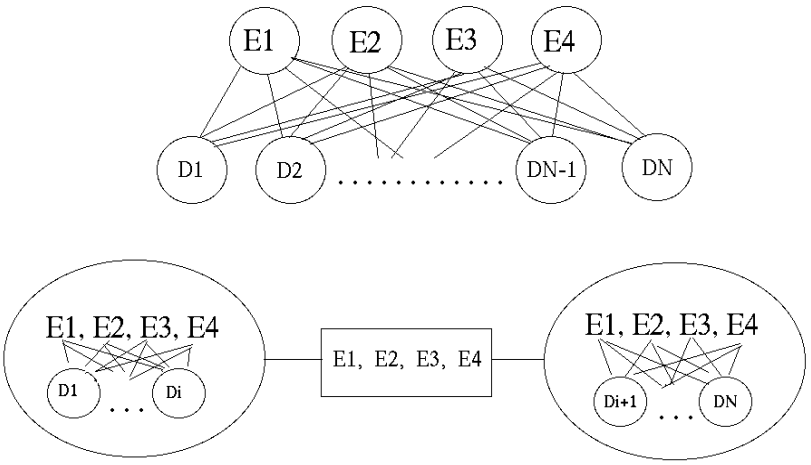

Figure 3.1: A fully connected BN and its resulting MSBN

Another approach is to approximate the network, which can be seen in Shen and Lesser [2]. In their paper Shen and Lesser suggest splitting up a BN by events, and then using the prior probabilities to marginalize the data. To accomplish this, they simply created a new BN for each event and all data connected with it (figure 3.2). To eliminate the effects of the other events and data on the current BN they used the prior probabilities from the original BN to marginalize out the effect. As shown by Carver and Haan [4] this approach performs poorly as the systems become larger and more connected. For example

2-4 non-fully connected BNs performed with an average accuracy of 98 percent, whereas a fully connected 6-12 BNs performed with an average accuracy of only 31 percent. This degradation is due to over approximation. The system they chose for decomposing the BN's does not take into account the strong bonds between much of the events and data.

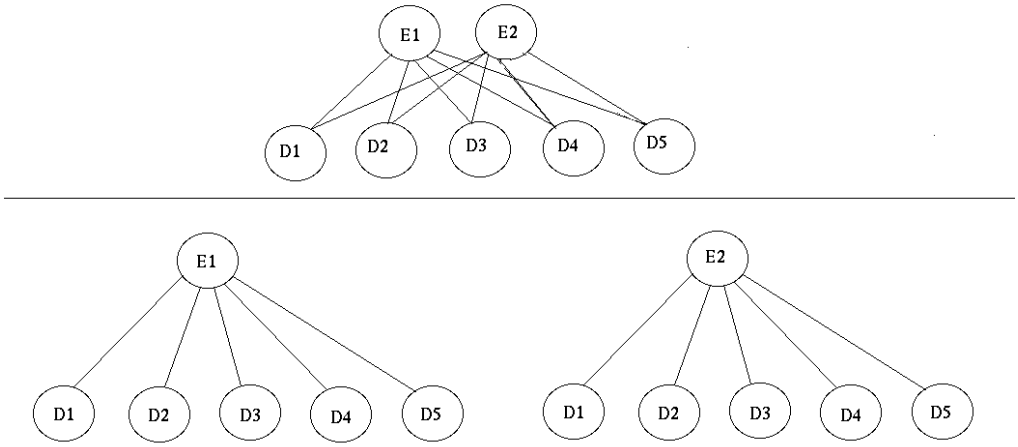

Figure 3.2: Fully connected BN and resulting Event BNs

Our goal was to create a system that could effectively use approximation, while still achieving accuracy close to that of the MSBN systems. We wanted to create a system that could be used to decompose a BN representation of an SI problem. The decomposed subnets could then be worked on by individual agents (or groups of agents) with a minimum amount of communication. The result would be a network that is an effective approximation of the original network, but with a faster time to solution and low communication needs.

3.2 IMPLEMENTATION DIFFICULTY

Care must be taken in developing algorithms for finding appropriate decompositions, since "brute force" approaches to finding optimal decompositions are exponential. In our model of an SI system we have a two layered BN with events in the top row that have causal links to sensor data in the bottom row. When analyzing properties of such a system, one must notice that there are 2^d possible data combinations and 2^e possible events combinations. So traversing over every possible combination quickly becomes infeasible as the system grows larger. This means that any techniques that are used must either not look at all these possible combinations, or have a potential to be sampled or approximated.

3.3 CODE METRICS AND SYSTEM SPECIFICATIONS

Code was written to generate, decompose, and assess BNs. All system code was written using the Common Lisp language and run under Steel Bank Common Lisp for the Linux platform. Analysis code was written in Common Lisp, and Perl under Linux. Also a Bayesian network library previously created for simulation and analysis in Carver and Lesser [1], was used as the base for the system. All other code was original and written by the author. The original code consists of approximately 1000 lines of Common Lisp code in 24 functions and 4 files, and 290 lines of Perl code in 4 files. The whole system was run

on two 64bit AMD machines running Fedora core 5.

3.4 EXPLANATION OF THE SYSTEM

3.4.1 System Model

The original SI model that our system takes in as input is represented as a BN (figure 2.1). This model consists of two levels, a level of events, and a level of data. The events in this model have causal links with the data. The target model is a type of DBN similar to a 2-agent MSBN. It consists of two individual subnets divided by a sepset (figure 3.3). All communication between the two subnets is done via updating the sepset probabilities. However, our network is not decomposed by finding variables that d-separate the subnets as this is far to inflexible. Instead, we use a calculation we have termed *weighted average conditional mutual information* (WACMI), which was presented in section 2.2.3.

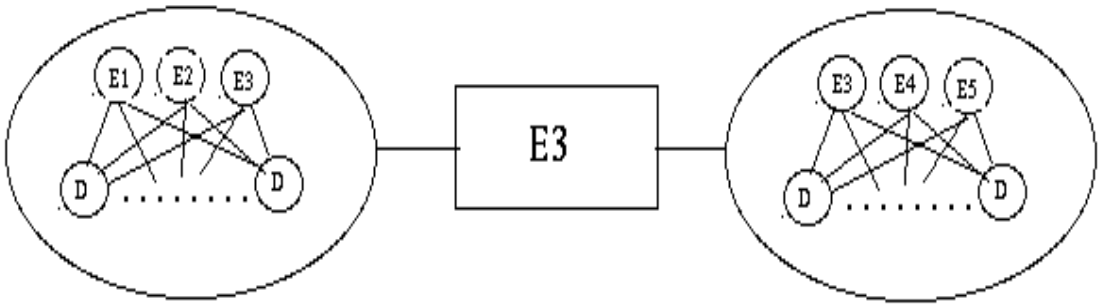

Figure 3.3: Junction Tree style model used after decompositions

3.4.2 Why Use Weighted Average Conditional Mutual Information

The concept behind using WACMI is this: subnets that have low WACMI should have little effect on one another. As a result the subnet/interpretation subproblems should be able to be solved independently, or with little communication. MI is the amount of information one set of variables give about another. In a system using d-separation the MI of the two subnets would be zero if knowledge of the sepses was known. Hence why systems base on d-separation have many of the properties they posses. However, unlike d-separation we are not limited to one decomposition, we are free to choose any decomposition that has a low MI.

Our system cannot use the simple MI (formula 2.1) to divide the BN, as the events in a two level SI model are independent without knowledge of the data. This means that we must look at the MI between events conditioned on the data (formula 2.2) so CMI is required. We must also take into account the fact that there is not just one possible data combination that affects the MI between events. In doing so, we must take a weighed average of the CMI over all of these data combinations. The purpose of WACMI (formula 2.3), is to give an overall picture of the amount of interaction between two sets of events.

3.4.3 Generating Bayesian Networks

Decomposition techniques were tested on SI-type BNs with appropriate properties,

which were randomly generated. Two styles of BNs were used, joint BNs and independent BNs. Joint BNs have an arbitrary *conditional probability table* (CPT). Independent BNs are different from joints in that they use a noisy-or model for their CPT. This model reduces the number of probability needed to represent the BNs. Independent BNs represent a situation in which each event can independently cause a piece of data to have a certain value. To generate joint BNs we used a function that took as input, the size of the BN that need to be generated, the minimum allowed value for interpretability, and the maximum number of trials. An example input would be 5-10, 90, where the first set represents the number of events and data, the second number represents the minimum accepted value for interpretability. The function then randomly generates BNs most BNs generated are fully connected. Fully connected means that all events are connected to all data. Then, once the BNs are generated the function assesses their interpretability. If the value is above the given minimum the function then tests the monotonicity of the BN. At this point it writes the BN out to a file named based on its interpretability and monotonicity. Independent

After all the BNs are generated one final analysis is performed. A function is run that tests all the BNs decomposability. The decomposability is written into a file, and BNs with a low decomposability (< 80 percent) are discarded. The resulting BNs have a high (> 80 percent) interpretability, and decomposability.

3.4.4 Determining the Best Decomposition

The steps described next can be seen in figure 3.4. To decompose the system we first split the events into two groups, looking for the split that has the smallest WACMI. The resulting event groups are the start of our subnets. Next we look at all the possible data groupings with the previously mentioned event groups, and pick the grouping that has the highest CMI. The reason we choose the highest CMI is because large CMI values mean that the data has a strong correlation with those events.

Now that we have two subnets each comprised of some portion of the data and events, we must choose the best sepset. The sepset group is chosen by looking at all possible event groups up to a given size, and finding the group that has the highest WACMI with both subnets. This group is the sepset.

All members of the sepset must be contained in both subnets, so they must be added to the subnets any subnet that does not contain them. The size of the sepset is limited due to the fact that we only want a small portion of the events in the sepset. More information on the exact s used for the decomposition process is listed in section 3.5.

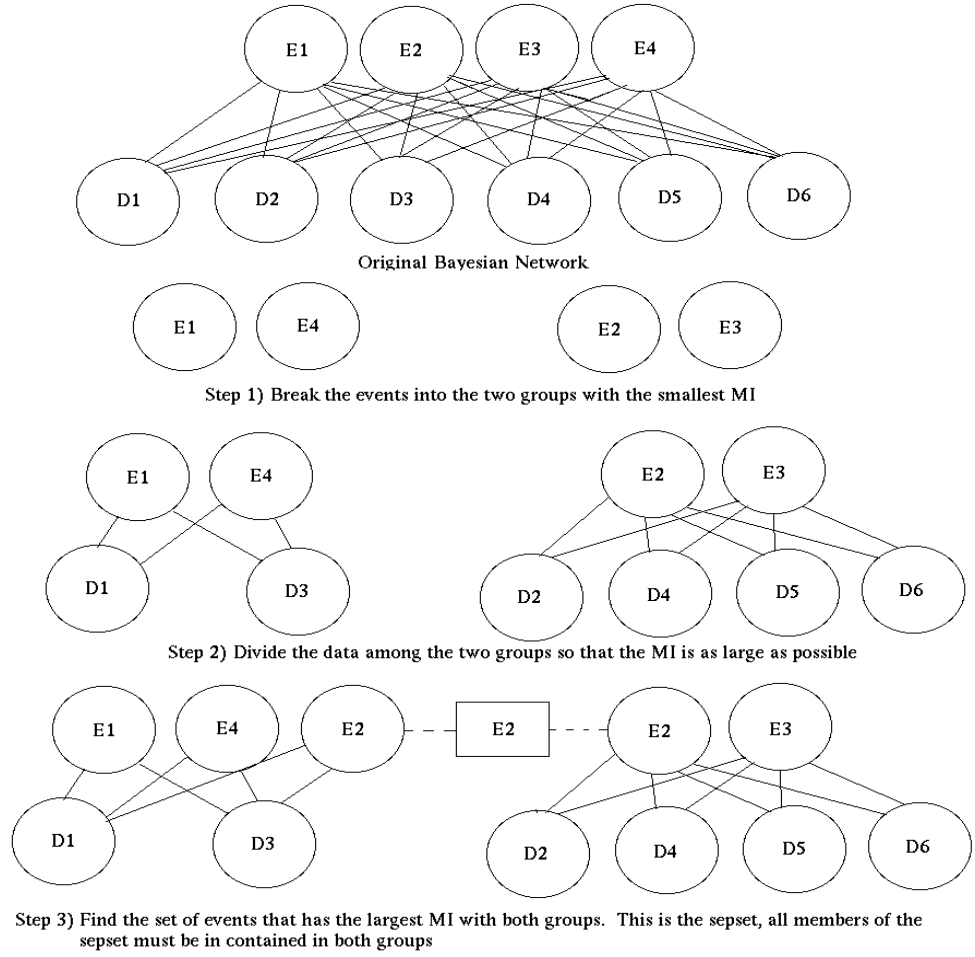

Figure 3.4 Steps of WACMI Decomposition

3.5 ALGORITHM

3.5.1 Conditional Mutual Information & Maximum Mutual Information

When calculating CMI (algorithm 3.2) it is possible that P(X,Y | C), P(X | C), or P(Y | C) could be zero. In the first case the value for MI cannot be computed, as we would

have the log of zero. In the last two cases we have a division by zero error. In all three cases we need anther calculation for MI. This is where the *maximum mutual information* (MMI) comes in. The maximum value for MI is determined by the smallest number of bits in either set, and is then further limited by the probability of that set occurring. The largest amount of information a group of variable can convey is the total number of bits it can represent. Also when calculating MI each of the sets in the summation is weighted by the probability of that set occurring. This is the reason MMI is calculated the way it is. MMI is represented in algorithm 3.1 in the figure X and Y are the groups of events that we are trying to determine the MI for, C is the set of data those events are conditioned on, and Bits(X) returns the number of bits represented by the set X.

input: X, Y, C
return Minimum(P(X | C) Bits(X), P(Y | C) Bits (Y))

Algorithm 3.1: *Maximum Mutual Information* (MMI)

Conditional Mutual Information (CMI) is needed in the computation of WACMI and for the decomposition of our system. CMI is calculated by computing the MI using conditional probabilities. CMI is algorithm 3.2, X and Y are the events, and C is the set of data on which the probabilities are conditioned.

```
input: X, Y, C
MI = 0
for all x in X
  for all y in Y
       if P(x | C) = 0 or P(x | C) = 0 or P(x,y | C) = 0
                MI = MI + MMI(x,y,C)
       else
                MI = MI + P(x,y | C) * log2 {P(x,y | C) \  [P(x | C) * P(y | C)]}
  return MI
```

Algorithm 3.2: *Conditional Mutual Information* (CMI)

3.5.2 Weighted Average Conditional Mutual Information

Weighted Average Conditional Mutual Information is the most important measurement in our system. Looking at its algorithm (algorithm 3.3) it looks easy to calculate, however, it must be noted that for every possible data set, c, it also makes a call to CMI which then runs through every possible set of X and Y. This makes the computation for WACMI very costly meaning we will want to reduce the use of the formula or find a

```
input: X, Y, C
WACMI = 0
for all c in C
       WACMI = WACMI + P(c) * CMI(X,Y,c)
return WACMI
```

Algorithm 3.3: *Weighted Average Conditional Mutual Information* (WACMI)

way to perform sampling on it. More on sampling WACMI is discussed in Chapter 4.

3.5.3 WACMI Two Agent Split With Sepset

The core algorithm in our system is algorithm 3.4. This is the algorithm that when given a BN, will return the best possible split and sepset. This algorithm works by looping over all possible event and data combinations finding the two subnets that have the minimum MI. It will also determine the event with the highest MI among the two groups and return that event as the sepset. With this information the original BN can now be split and tested.

3.6 WACMI Two Agent Split Variations

3.6.1 Unused Variations

When first designing the algorithm to split the data using WACMI, we tried different variations on measuring the MI. One issue that was the most trouble was determining the best split for the data. There are many different ways the MI for computing the data split can be measured. The first variation we considered was splitting the data in the same fashion as we divided the events. However, after the split has been determined we then calculate which of the two groups of data have the highest MI with the first group of events. These two are then grouped, and the only remaining choice for the rest of the data is to be grouped with the other set of events. This variation has the advantage that

```
input: B, S
E = B->Events     D = B->Data
NumE = 2^E-1      NumD = 2^D-1
BestSplit = 0
CurWACMI = 0
MinWACMI = highest positive number
{Determine least mutual event split}
from i = 1 to NumE / 2
        SubE = the subset of elements in E represented by i in binary
         -SubE = the elements of E not in SubE
        CurWACMI = WACMI(SubE ,-SubE, D)
        if CurWACMI < MinWACMI
                BestSplit = i
                MinWACMI = CurWACMI
{Determine most mutual data split}
MaxWACMI = lowest negative number
CurWACMI = 0
BestDataSplit = 0
SplitE = the subset of elements in E represented by BestSplit in binary
-SplitE = the elements of E not in SplitE
from j = 1 to NumD
      SubD = the subset of elements in D represented by j in binary
     -SubD = the elements of D not in SubD
      CurWACMI = WACMI(SplitE, SubD, NIL) + WACMI(-SplitE, -SubD, NIL)
                if CurWACMI > MaxWACMI
                BestDataSplit = j
                MaxWACMI = CurWACMI
{Determine the most mutual one element sepset}
CurWACMI = 0
BestSepset = 0
MaxWACMI = lowest negative number
SplitD = the subset of elements in D represented by BestDataSplit in binary
-SplitD = the elements of D not in SplitD
for all e in E
        CurWACMI = WACMI(e, SplitE, SplitD) + WACMI(e, -SplitE, -SplitD)
        if CurWACMI > MaxWACMI
                BestSepset = e
                MaxWACMI = CurWACMI
reuturn SplitE, SplitD, BestSepset
```

Algorithm 3.4: WACMI 2 Agent Split With Sepset

only one of the event groups must be looped over. This variation was not chosen as it did not yield good results. This is due to the fact that the second subnet gets stuck with the leftover data that is not matched to the first subnet. This means that it often times does not perform as well.

Another variation we considered for measuring the MI of the data split was to take into account the MI between the data sets. To achieve this, you must loop over all possible data combinations and compare the MI of the data and events just as in algorithm 3.4. However, to perform this calculation you must subtract the MI between the data sets from the current value for MI. The advantage of this method is it results in a more accurate value for the MI of the subnets. This is because the current method does not look at how the two data groups might affect one another. The disadvantage of this variation is it requires computation of the WACMI between the two data groups. We decided to not use this variation as it did not give a notable improvement to the overall results, and caused a significant increase to the runtime of the algorithm.

Yet another variation we considered for measuring the MI of the data split was to condition the calculations over all of the data. The current method only conditions the WACMI of the data/event groups over all possible values for the data in that group. The reason we decided not to use this method is because, once again, it did not add a significant increase in results. However, the method does increase the runtime of the algorithm.

3.6.2 Used Variations

The main variation to the algorithm that is currently in use and will be analyzed in chapter 4, is finding the best even event split. This variation is necessary since often times the best split will only separate off one of the elements. This is not a large enough split to be practical in larger systems. This is a simple variation to the algorithm and can decrease the run time as it reduces the number of sets that must be looked at. To accomplish this algorithm 3.4 must be changed in the following way. When looping through all the sets of data analyze only the sets that are of approximately equal size. Then the resulting output will be the best equal split of the BN.

Another variation that is also in use is adding the possibility for larger sepsets. This was needed when we began to test larger BNs as a one element sepset is not sufficient. To accomplish this, the algorithm, when choosing sepsets must loop over all possible event sets of the given size or smaller. This increases the runtime of the algorithm as it must now loop over sets as opposed to individual events. However, this is a necessary evil for larger BNs.

3.7 WACMI Issues

When computing WACMI we came across two problems that were not well defined in the literature. The first problem, is the question, what is the maximum value for MI?

The second problem occurs when any of the probabilities in the calculation for MI become zero. The first problem occurs when the two sets are identical. This problem was easy to solve upon realizing that the largest amount of information any set can convey is its size in bits. The second problem is a bit more difficult, when taking the log of infinity or dividing by zero the results are undefined. The choices are clear, the value will either be the maximum possible value, or zero depending on which grows faster in the formula. We solved this through empirical means, running the system with both values and testing which performed better. Overall, we determined that setting the value to the maximum was best. This generated better results.

3.8 TESTING DECOMPOSITIONS

3.8.1 Splitting the Bayesian Networks

After generating the BN and running the WACMI split algorithm and its variations, we now know our best and equal problem decompositions with and without sepsets. We must now split the BN, using the the outputs from the above algorithms. The outputs are as follows: best split, best split with a sepset, best even split, and best even split with a sepset. Each of the outputs is then fed into an algorithm that splits the BNs into two files, adding the sepsets to both BNs. The events and data are written to their corresponding files using their marginal probabilities.

3.8.2 Output Files

After generating the files and computing the decomposability, monotonicity, and interpretability the result is three files. One file contains the BN, one file contains the decomposability of the BN, and the final file contains the monotonicity and interpretability. Then the BN is run through the WACMI decomposition algorithm and then the splitting algorithm. The result of this is an additional eight files. These eight files consist of the the BNs for agent one and agent two, for all four types of splits. The four types, again, are best split, best split with a sepset, best even split, and best even split with a sepset. The format of these files can be seen in figure 3.5. Contained in each BN file is all the information needed to represent the BNs for each agent. At the top of the BN file it shows what events and data the agent is responsible for. The files also contains the prior probabilities of the events and the conditional probability tables for each of the pieces of data.

3.8.3 Testing Performance

To test the performance of our decomposed BNs we used a function that emulated a two agent junction tree style DSI (figure 3.3). First each agent processes its local data e.g. the data specified in its subnet. Second the resulting event probability vector is passed via the sepset to the other agent, simulating communication. This is the join tree model for message propagation. Third each agent integrates the received vector message. This is

done by multiplying the vector's elements with the corresponding elements of the agent's event probability vector. Fouth the agents determine the best interpretation for each event in there subnet. In interpreting each event, the event is considered to have occurred if its resulting (conditional) probability is greater than 50%, else it is considered to have not occurred. Fifth, the resulting values for the events are compared to the values for the same events in the MAPI given all the data. The probability that the system would produce the same event interpretations as the MAPI is then returned as the overall performance of the system.

```
S
(E3)
(D6 D7)
(E3)

joint
S (E3 .01 )

joint
D6 (E3) (.19 .41 )
joint
D7 (E3) (.38 .72 )
```

Figure 3.5: Sample BN file for an Agent

CHAPTER 4

ANALYSIS

4.1 MANAGING DATA

The results of running our system on over 1600 BNs, each resulting in 14-16 files, was over 22000 data files that had to be analyzed. To begin to comprehend what was going on with all of this data, a system was needed to sum up the information. To this end three Perl scripts were written.

The first Perl script took two inputs, a percentage to be ranked as good, and a percentage, below which, the decompositions was ranked as bad. The values that we used were 90 percent and 80 percent. Any decomposition that performed from 80-90 percent was ranked as average. The script then went through every data file in a given folder recursively and summed the data into four files.

The first file is a count file that gives the overall performance percentages of the data. The other three files are labeled good, average, and bad. They contain in a shorthand format the corresponding data for all the BNs that were ranked in that category. Examples of these files can be seen in figure 4.1 and 4.2. In figure 4.2 we can see that each BN is placed in the file labeled with its location, and contains both normalized and unnormalized

values for all four of the different decompositions.

```
Total number of networks: 916
Good >= 90%
Bad <= 80%
Good count: 686
Good percent: 74.89%
Average count: 151
Average percent: 16.48%
Bad count: 79
Bad percent: 8.62%
```

Figure 4.1: Example of the Count Summation File

```
/cdps/mynetworks/Fconnected/6-12 (()()()()()())/j6-12-19-output.data
bs (0.9109014913056876d0 0.9110837041950143d0 0.9998000042273968d0)
bss (0.9063995569898358d0 0.906580869331225d0 0.9998000042273968d0)
es (0.7515419744922072d0 0.7516923097764608d0 0.9998000042273968d0)
ess (0.7575302328159917d0 0.7576817659661635d0 0.99980000042273968d0)
```

Figure 4.2: Example of one BN from the Good Output File

The next Perl script was designed to recursively parse a folder and convert all the data files into one flat file containing a human readable format. An example of this results file can be seen in figure 4.3. Using this file it is now easy to look at all performance values in one place and compare them to one another.

```
folder              || name   || type || decomp || monoto || interp  ||  bs    ||  bss   ||  es    ||  ess   ||
search-joint-7-14-1 || j7-14-2 || joint || 0.9784 || 0.8285 || 0.9209 || 0.8828 || 0.8797 || 0.7456 || 0.7601 ||
search-joint-7-14-1 || j7-14-1 || joint || 0.9817 || 0.8426 ||  0.9241 || 0.7296 || 0.7334 || 0.7859 || 0.7847 ||
```

Figure 4.3: First Three Lines of a Results File

The last Perl script was designed to allow the information from the results files to be entered into a spreadsheet application. The Perl scripts input was a results file, it then

converted the results file into a comma delimited list containing the exact same information. The new comma delimited file was then entered into a spreadsheet to allow for further analysis and the creation of graphs.

4.2 PERFORMANCE

4.2.1 Summary of Performance

Overall, the decomposition method performed well. Of the 916 randomly generated, BNs ranging in size from 2-4 to 7-14, 91.74% had an accuracy at the average level (80%) or higher. The break down is as follows: 686 BNs (74.89%) performed with an accuracy higher then 90%, 151 BNs (16.48%) preformed with an accuracy between 80% and 90%, and only 79 BNs (8.62%) performed with an accuracy below 70%. Looking at all 1646 BNs, including those that do not have high values for decomposability, monotonicity, and interpretability, 84.26% performed with an accuracy of 80% higher.

4.2.2 Best Split vs. Even Split

The more important value to analyze is even split with a sepset, as it is the most useful. Looking at figure 4.4 we can see that, once again, the majority of the decompositions performed above 90% and very few BNs performed less than 80%. Figure 4.4 is a graph of sorted even split and corresponding best split decompositions with sepsets. The BNs range from 2-4 to 7-14 in size. Larger networks will not be possible until

Figure 4.4: A Graph of Best Split vs. Even Split Values
Sorted By Even Split

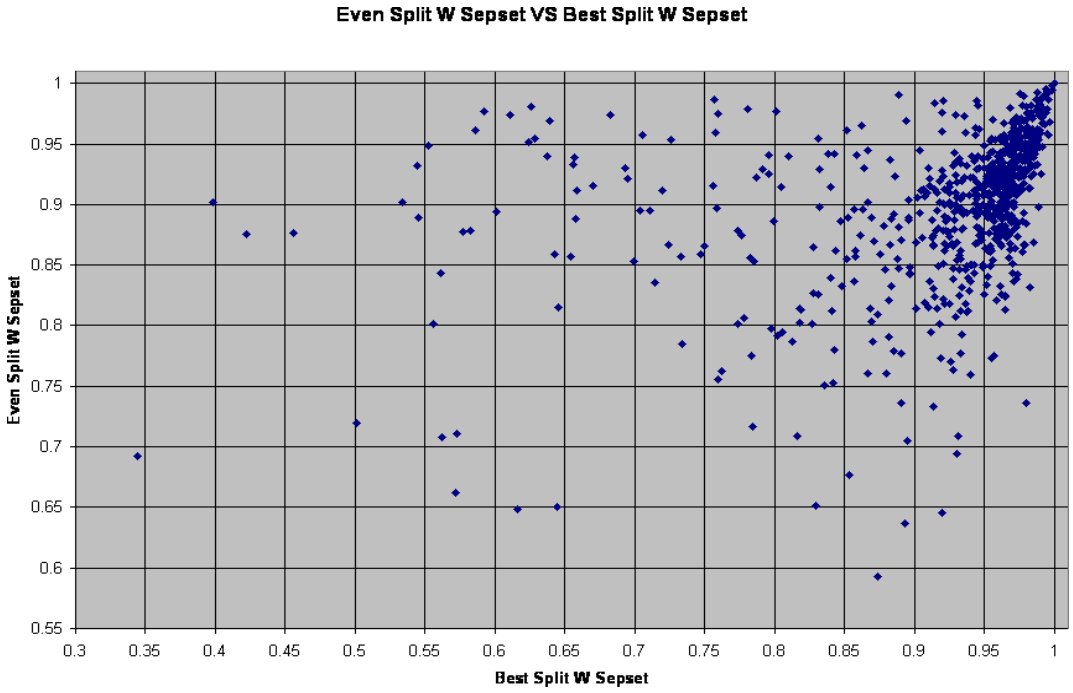

Figure 4.5: Even Split vs. Best Split (with sepsets)

sampling is introduced. Sampling is discussed more later in the paper.

Figure 4.4 also shows the relationship between best split and even split. Best split in almost all cases performs better then even split. Figure 4.5 shows this relationship in another way graphing each BNs even split and best split. As you can see, they both follow the same general upward trend. However, this graph makes it easier to see that although the DBNs resulting from best split, on average, perform with a higher accuracy, they have many lower values. This is in contrast to the DBNs resulting from the even split which have no accuracy's lower than 58%. The reason that even split DBNs have less deviation is due to the fact that the data and the groups are evenly spread among the two subproblems. In the best split case, many times one of the subproblems will have significantly less data then the other. The lack of data causes that subproblem to perform with a lower accuracy, the resulting poor performance then brings down the accuracy of the whole system.

This is what we would expect to see as best split DBNs will have the same or higher MI value, thus, should result in a better overall accuracy. However, even split decomposes the system in such a way as to evenly distribute the data. This allows both subproblems a significant portion of the data, allowing for less varied results.

4.2.3 Sepsets vs. Non-Sepsets

Sepsets overall did not greatly improve the accuracy of the system. This was unexpected as they allow for communication, i.e., more accurate probabilities. Figure 4.6

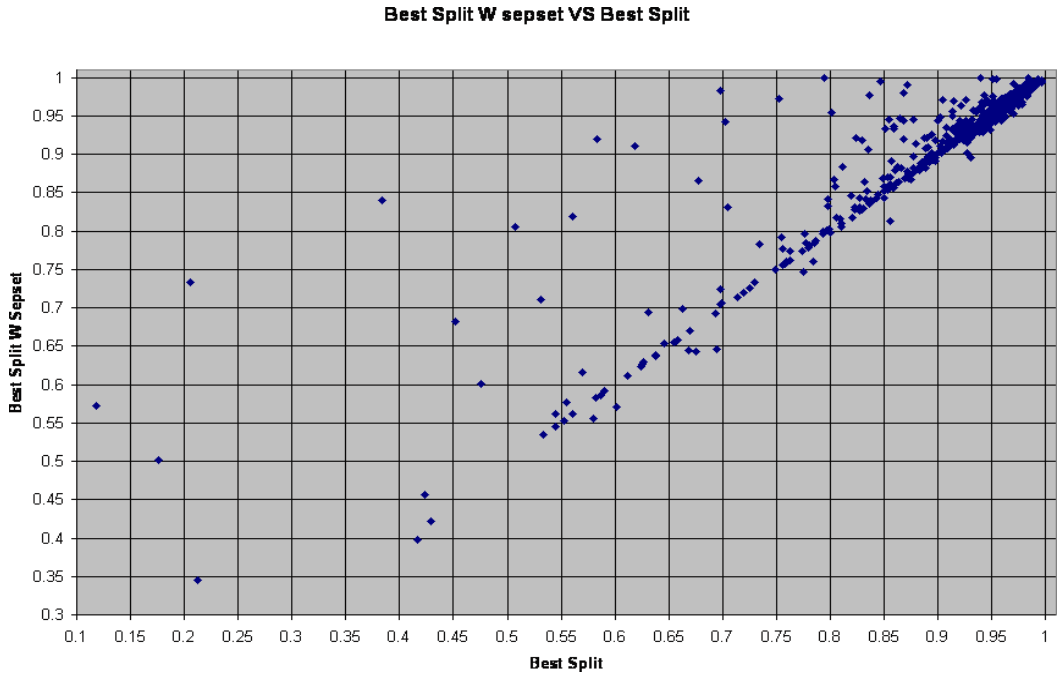

Figure 4.6: Best Split W Sepset vs. Best Split

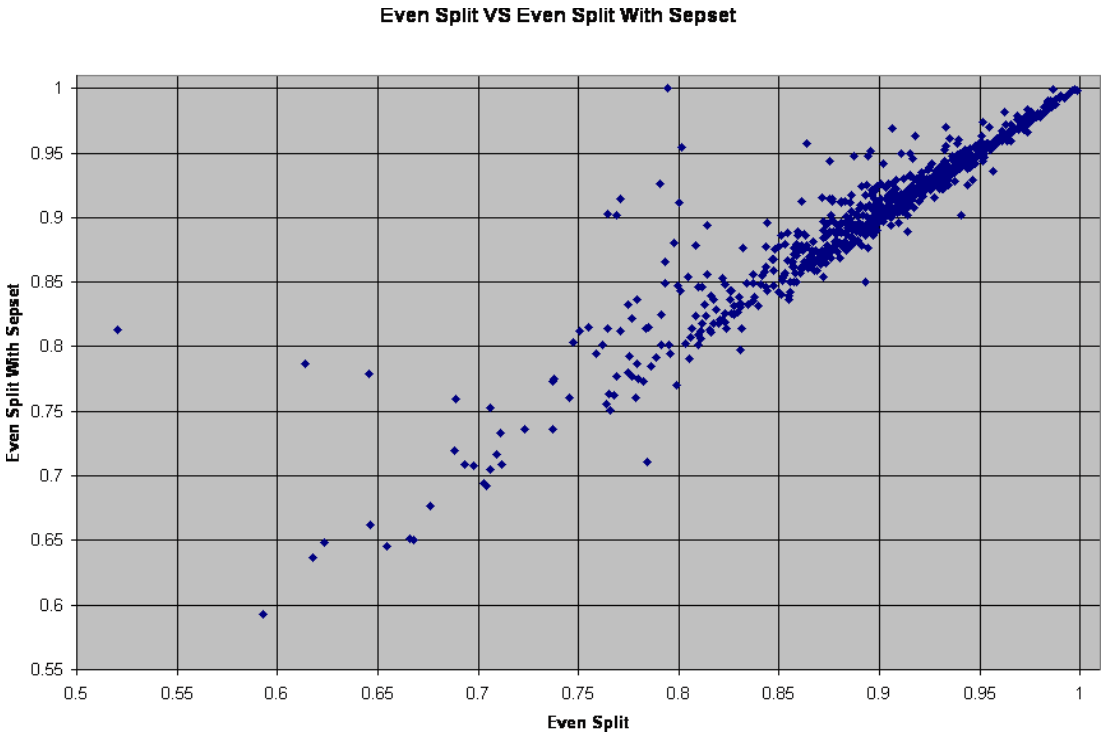

Figure 4.7 Even Split W Sepset vs. Best Split

shows this, in the figure we can see that best split with a sepset in almost all cases performs almost equal to the case without a sepset. Sepses do help some, if we note the range of each of the measures we can see that best split has values as low as 12% accuracy where, when a sepset is added our lowest value becomes 34%.

If we look closer at figure 4.6 we see that a lot of the values fall on the horizontal. This means that adding a sepset did not improve the accuracy. This is most likely caused by two factors. The first factor is how the best split is divided, often times most of the events and data end up in one of the subnets. The larger subnet in the system, affects the performance of the system to a greater degree, simply because it has more events to predict. When a sepset is added in this situation it does not greatly increase the amount of data affecting the larger subnet. Because of this, it does not affect the outcome to a large degree. If we look at figure 4.7 (Even Split) we can see that adding a sepset affects the outcome to a larger degree than in best split. This result is because each subnet has only half the data, unlike best split. Thus connecting them adds a larger amount of information about the data to each subnet, and this causes the accuracy to increase to a larger degree.

The second factor is sepset size. The fact that adding sepsets does not greatly improve the results may mean that the sepsets are too small. Larger sepsets allow for more information to be passed between the subnets. Thus, increasing the sepsets in future trials may improve these results. In general these results mean that we decomposed the system in

an effective way almost eliminating the need for communication. However, this result could be bad as in the future it will almost certainly be necessary for communication.

4.2.4 Size

Size is another factor that affects the performance of our system, as BNs grow larger their interactions grow more complex. Figure 4.8 shows a graph of the average best split and even splits with sepsets performance graphed against the size of the BN (events+data). Figure 4.8 shows that as size increases average performance decreases. The graph shows a major decrease in performance as the size increases. This, however it is not quite this simple. There is one major issues to note with the graph in figure 4.8. There is significantly more data available for the BNs of sizes 9 through 14 than of the sizes at either end of the graph. This means that the values in the middle show a much more accurate picture then the end values. The reason for this is 2-4 BNs or size 6 are too small to tell us much information, while BNs of size 6-12 and 7-14 (size 18 and 21) are too time consuming to run in large quantities.

While there is a downward trend, it is not quite as drastic as figure 4.8 shows. If we look figure 4.9, we can see all the data points graphed not just the averages. Figure 4.9 shows all the data sorted by size, i.e., the size of the BNs increase in groups from left to right. The graph does not show the sizes of the groups, however this is not important for what we are trying to discern. Looking at the graph we can see that the majority of the

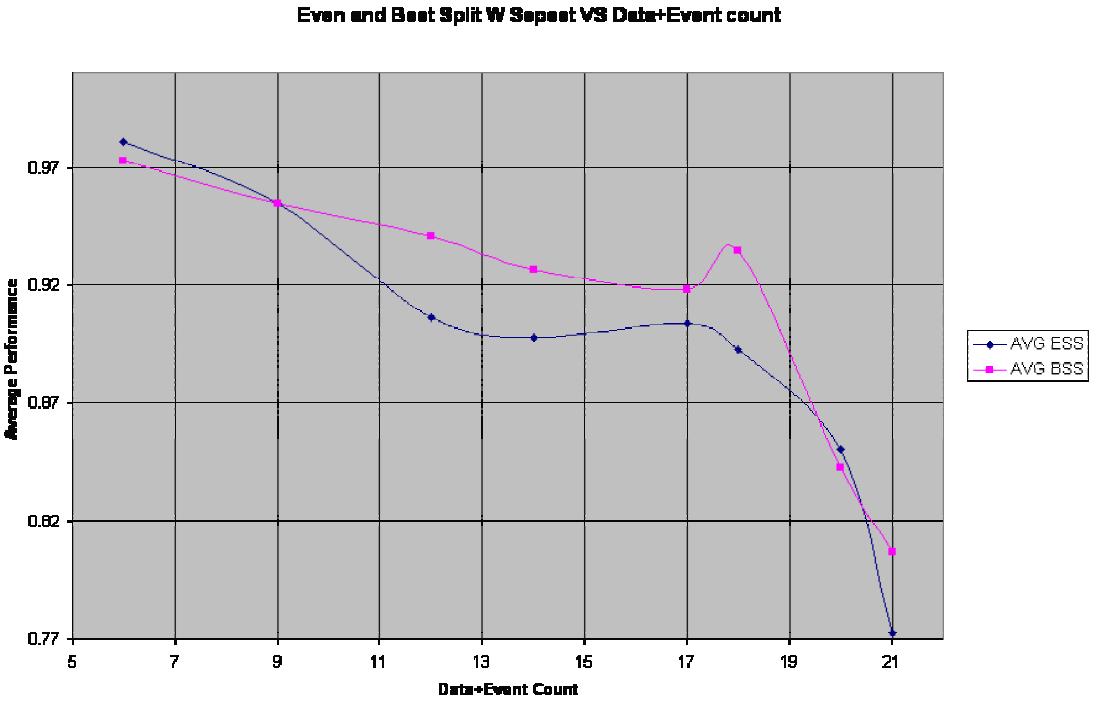

Figure 4.8: Best Split and Even Split with Sepsets vs. BN Size (Events+Data)

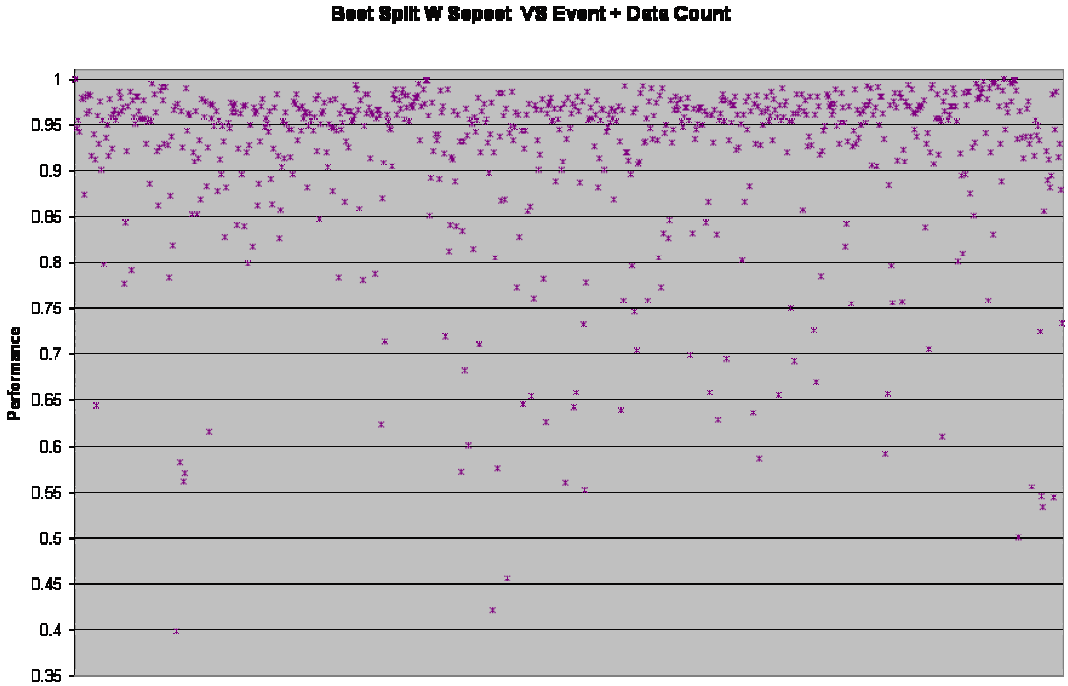

Figure 4.9: Best Split with Sepset Sorted by BN Size (Events+Data) Size Increases from Left to Right

decompositions performed with an accuracy over 90% across all sizes. This shows that although figure 4.8 is accurate in showing the downward trend, it is skewed by the number of BNs comprising the averages. This, for example, means that the 7-14s performance average is affected to a larger degree by its low accuracy values.

4.2.5 Effects of Monotonicity, Interpretability, Decomposability

Decomposability, Interpretability, and Monotonicity are correlated with performance. However, they are inherent to the BN model and cannot be changed to affect performance. Once these values approach 80%, they then appear to no longer have an effect on the BNs. If we look at figure 4.10 we can see a general upward trend in the performance as decomposability increases. Figure 4.10 shows a graph of all of the fully connected BNs not just those that have decomposability over 80%.

This same trend can also be seen with Interpretability, but it is not noticeable with monotonicity (figure 4.11). Figure 4.11 shows that as monotonicity ranges from 66% to 100%, no notable trend appears in the data.

If we only look at BNs that have values greater than 70% for Monotonicity, Decomposability, and Interpretability, we see no noticeable relationship between these values and the performance of the system. This lack of relationship is shown in figure 4.12. In the figure the best split with sepset data is sorted in ascending order, and then graphed vs. the corresponding monotonicity, decomposability, and interpretability.

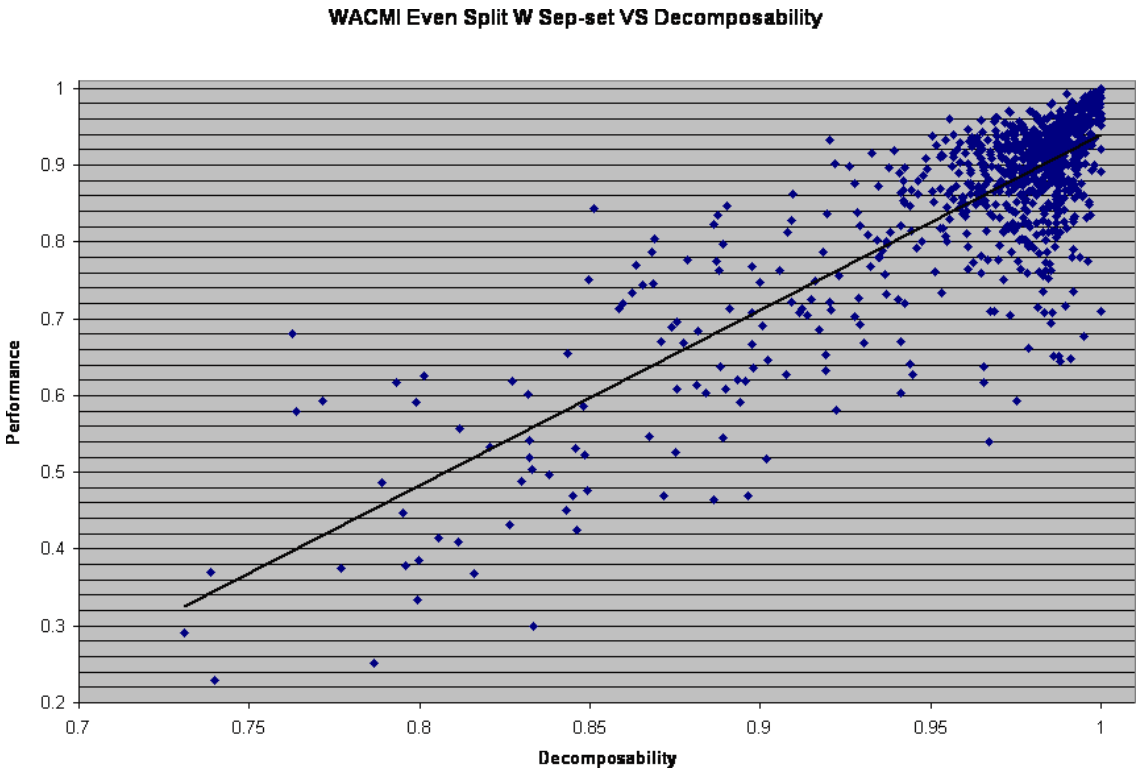

Figure 4.10: Even Split With Sepset Performance vs. Decomposability

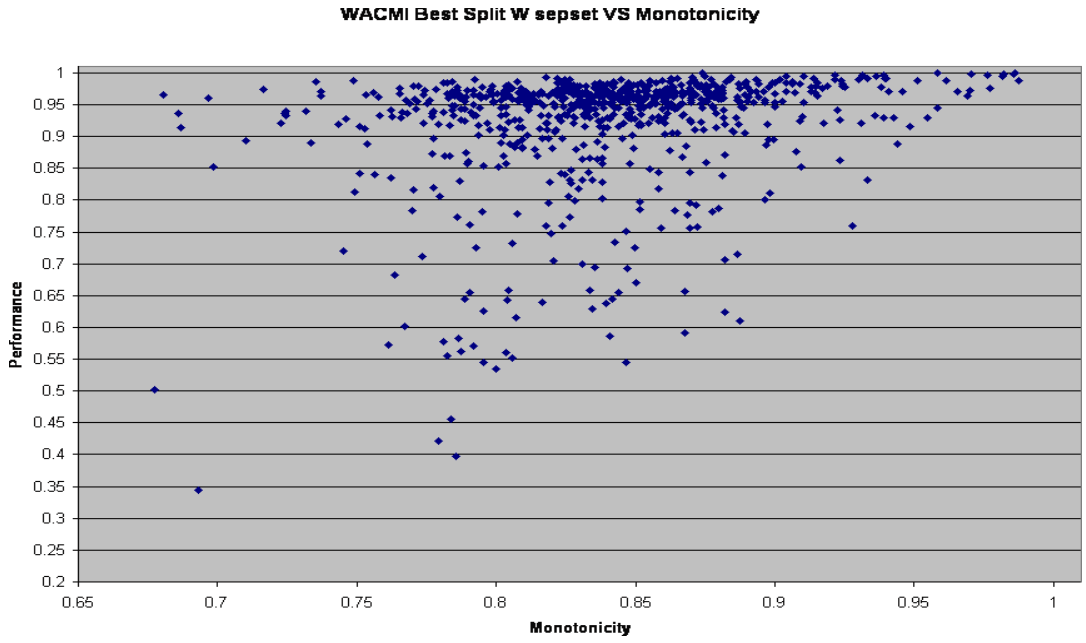

Figure 4.11: All BNs Best Split With Sepset vs. Monotonicity

This data in this section shows that although monotonicity, interpretability, and decomposability are important values, meaning they should be greater than 70%-80% for us to expect performance greater than 90% from our system. Once they are large enough, they are no longer the determining factor in performance.

4.3 IMPROVEMENTS OVER PREVIOUS WORK

Our method shows improvements over previous methods using approximation, considering the majority of BNs appear to decompose well using our system. Our

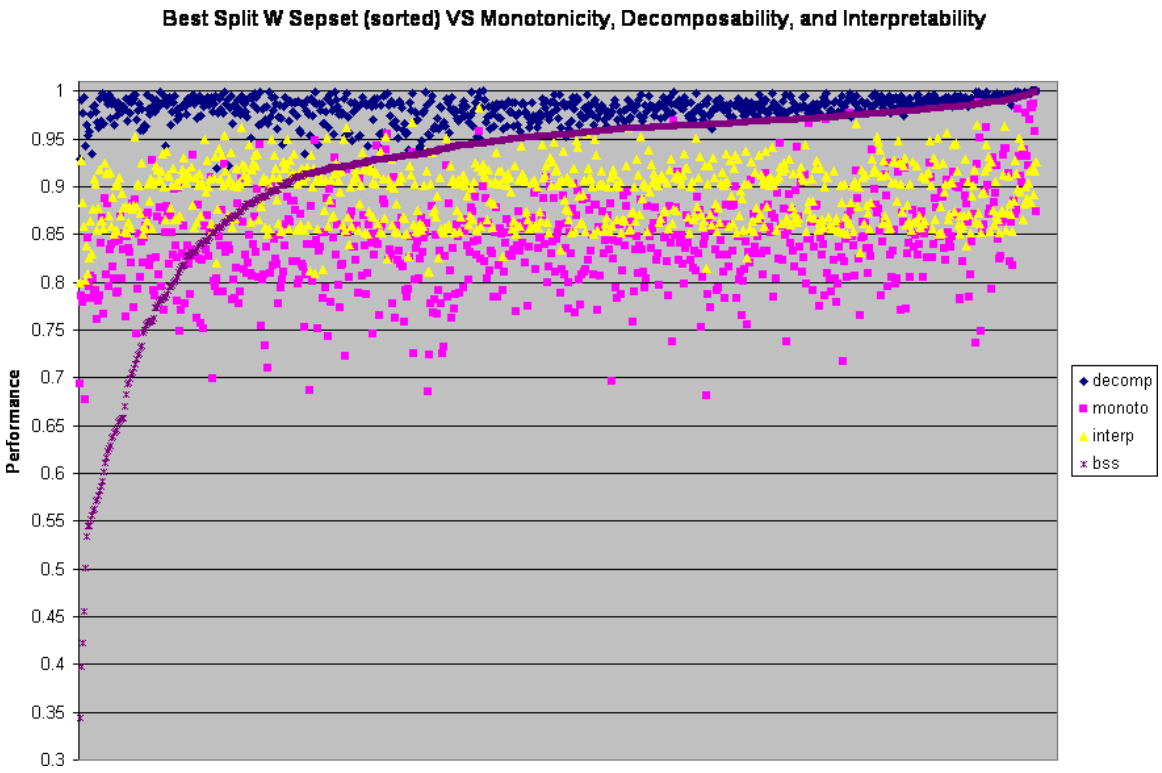

Figure 4.12: All BNs Best Split with Sepset (sorted) vs. Decomposability, Monotonicity, and Interpretability

decomposition method also appears to cope better as BNs grow larger and furthermore

does not degrade as we have seen in previous systems. Nothing can currently be said about BNs larger then 7-14 as they are to large to run. However, in future work we hope to adapt our techniques to be able to run on much larger BNs by using sampling.

Our method also shows some improvement over typical MSBN systems as it is more flexible. Our system can be used to decompose the system in different ways, the traditional d-separation method does not allow for this flexibility. These positives, however, come at a trade-off as our method uses approximation as opposed to the exact inference in MSBNs.

4.4 SPEED OF THE SYSTEM

A critical issue for our approach is speed. The calculation for WACMI is exponential in nature. This is due to the calculation computing the CMI of all possible values for the data and events, which grows exponentially as the BNs size increases. Also our algorithm looks at all possible event and data combinations when determining the best subproblems. Combinations grow in a factorial manor causing the other notable speed decrease in our system. We are hoping to remedy this through the use of sampling. This is discussed more in the next section and in the following chapter.

4.5 SAMPLING WEIGTHED AVERAGE CONDITIONAL INFORMATION

Weighted average conditional mutual information is an exponential function that quickly becomes infeasible as the number of data and events grow. To reduce this some

form of sampling is needed. Random sampling and then normalizing the result is the technique chosen to accomplish this goal. Normalization must be performed because we are trying to achieve a summation of the values and a sample will only represent a portion of this total. Thus, the sample must then be divided by the probability of the population it represents to make up for its proportion.

However, before any of this could be done, we first had to analyze WACMI to determine if its CMI values formed a distribution that could be randomly sampled. The reason this must be checked is due to the often random looking nature of BNs. Values for CMI or the condition probabilities may be random and it would then be almost impossible, using sampling, to get an accurate value for the overall summation of the *weighted CMI* (WCMI) values. The computation for WAMCI takes the CMI value and weights it by the probability of the conditioning occurring. In most BNs a small portion of the conditions make up most of the probability of the system as seen in figure 4.13. Figure 4.13 is a histogram of all the possible condition probabilities for a 5-10 BN, this shows that a small portion of possible data values or conditions make up the bulk of the probability and they form a distribution that should be easily sampled.

Next we must look at the distribution of the CMI values (figure 4.14). Figure 4.14 shows a histogram of all possible CMI values for one set of events in a 5-10 BN. This figure shows that the values fall into a normal looking distribution. This is not our ideal

distribution for our sampling, however, it is not random and as such should not skew our sampled values of WCMI. Lastly we must look at the a histogram of the WCMI values (Figure 4.15) to determine if our final sampling can be performed. Figure 4.15 shows that the WCMI values fall into a nice sharp distribution. This means that a somewhat small sample of the population should yield accurate values for WACMI.

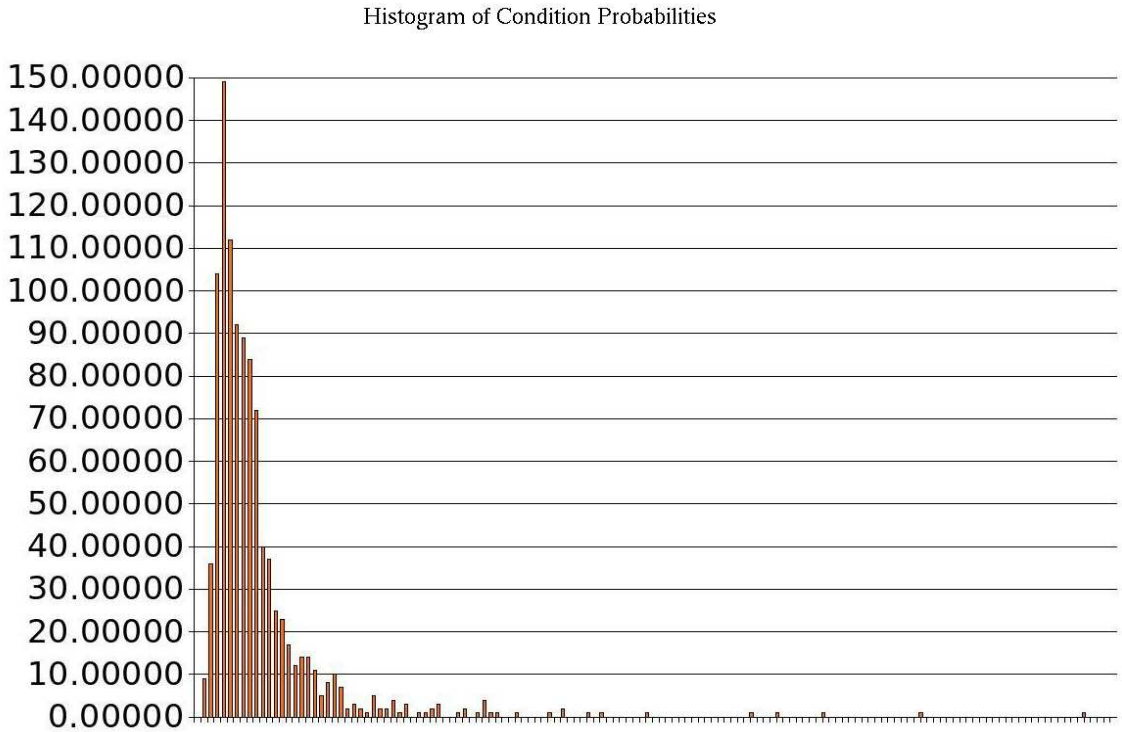

Figure 4.13: Histogram of Condition Probabilities
probabilities (X axis) range from .0002 to .014 by .0001 increments
Y axis represents a count of the number in each range

Based on these findings we then moved forward with random sampling. This changes the WACMI algorithm (algorithm 3.3). Using this calculation for WACMI with a sample size of about 15-20% yields consistent results that vary in value little across

multiple runs. Also using this calculation yielded the same BN decompositions as the original calculation for WACMI for all BNs that we tested it on.

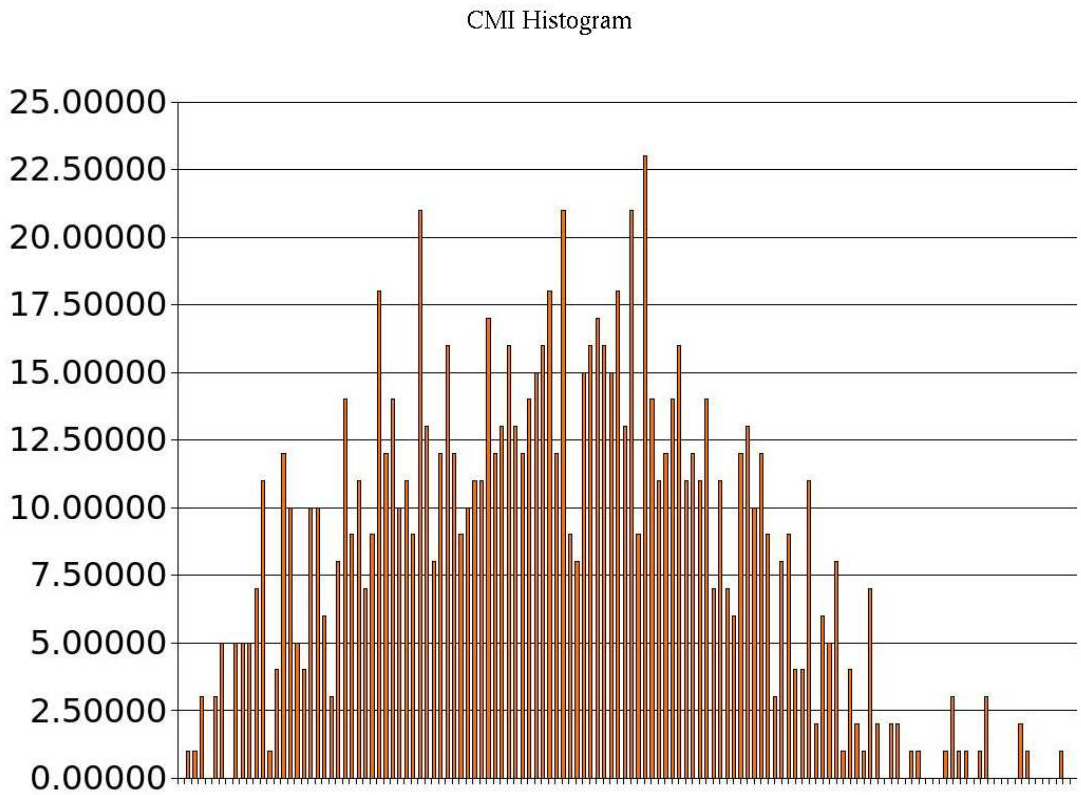

Figure 4.14: Histogram of CMI values
CMI values (X axis) range from .02 to 1.3 by .01 increments
Y axis represents a count of the number in each range

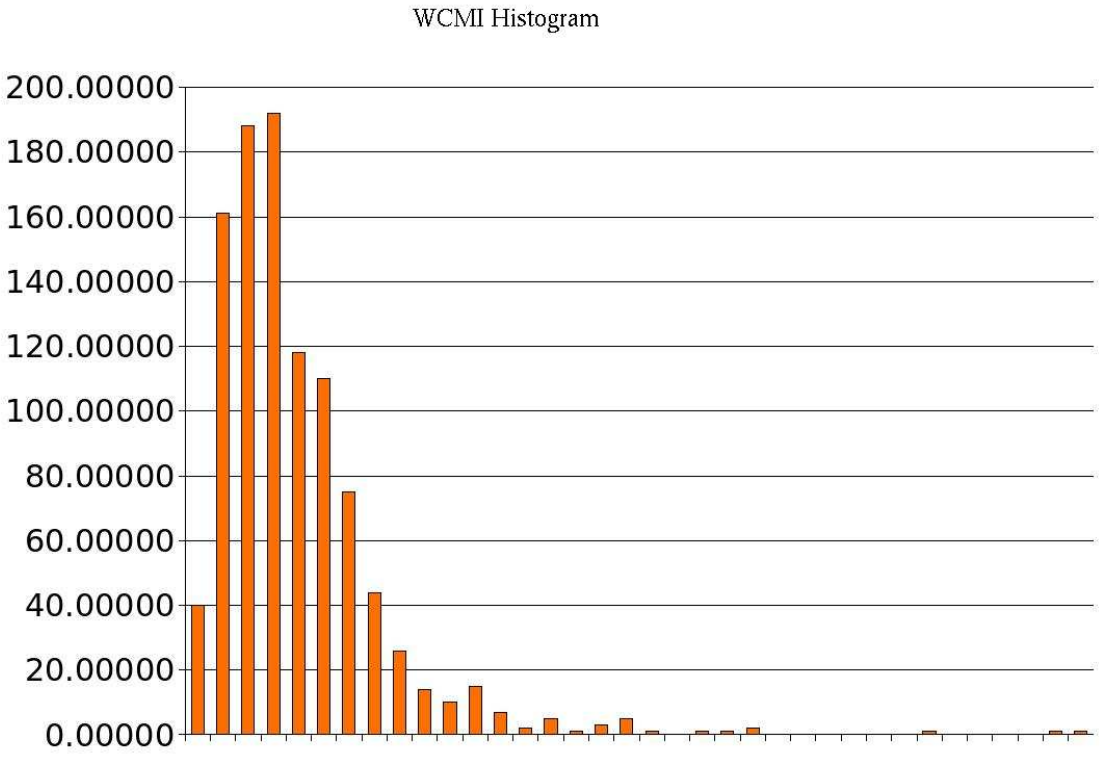

Figure 4.15: Histogram of WCMI Values
WCMI values (X axis) range from .0001 to .003 by .0001 increments
Y axis represents a count of the number in each range

```
input: X, Y, C, SampSize
WACMI = 0
TotProb = 0
NumSamp = 0
RandCond = NIL
while NumSamp < SampSize
        RandCond = a random value of conditions from C
        TotProb = TotProb + P(RandCond)
        WACMI = WACMI + P(RandCond) * CMI(X,Y,RandCond)
        NumSamp = NumSamp + 1
return WACMI / TotProb
```

Algorithm 4.1: Sampled WACMI

These results mean that sampling WACMI in this manner is an effective way to reduce the runtime of our overall algorithm.

The above method does decrease the run time for WACMI significantly, however, it does not cause the algorithm to be non-exponential. This is due to the exponential nature of calculating CMI. Because of this we must also sample CMI. CMI must take a summation of the values across all possible event combinations, this can be seen in algorithm 3.2.

We solved the problem with CMI in the same manner as WACMI, through random sampling and normalization. This changes the CMI algorithm as shown in algorithm 4.2. Sampling CMI in this manner yields slightly less accurate values and requires a larger percentage of the population then above (30-50%). However, using this still yields the

same results for our decompositions.

```
input: X, Y, C, SampSize
MI = 0
NumSamp = 0
Ratio = 2^(x+y) / SampSize
RandX = NIL
RandY = NIL
while NumSamp < SampSize
        RandX = random value of events from X
        RandY = random value of events from Y
        if P(RandX | C) = 0 or P(RandX | C) = 0 or P(RandX,RandY | C) = 0
                MI = MI + MMI(RandX,RandY,C)
        else
                MI = MI + P(RandX,RandY | C) *
                        log2 {P(RandX,RandY | C) \  [P(RandX | C) * P(RandY | C)]}
return MI * Ratio
```

Algorithm 4.2: Sampled CMI

The sampling approach to WACMI and CMI greatly reduces the run time of our algorithm. It also yielded the same decompositions as the original method on all BNs that we tested it on. This means that so far this is a viable method for calculating WACMI. In the future, however, this may become either infeasible or inaccurate as BNs grow larger and the proportion of the population that must be used becomes smaller.

CHAPTER 8

FUTURE WORK

8.1 CHANGES TO THE ALGORITHM

8.1.1 Changes To Increase Speed

One change to the algorithm that could greatly increase its speed is to change the way the data is decomposed. One way to decompose the data would be to look at the MI between individual pieces of data and the events. This speeds the algorithm up two fold. First this would eliminate the need to look over all combinations of the data, reducing this portion of the algorithm from factorial to linear time. Second, each calculation for MI would only have one piece of data, this would reduce the number of values that WACMI would have to parse.

Adjusting the way data is decomposed in this manner does increase the speed significantly, however, it is not as accurate of a calculation for MI. Because of this fact, this method may yield worse results.

8.1.2 Changes For Implementation With Larger Bayesian Networks

The system as it stands only decomposes the BN into two subnets that can be run in a two agent DBN type system. For larger BNs this is not a plausible approach. The

algorithm needs to be adjusted to break the system into more subnets. An even number of subnets seems to be a logical approach. For a 20-40 BN, decomposing it into four 5-10 networks seems reasonable.

The main difficulty in increasing the system in this fashion is determining what sets should be connected. Also, it becomes infeasible to compute the WACMI between all the groups as the number of possible group combinations grows in a factorial manner. Along this same line, computing the data decomposition is also impractical as we must now perform the calculations across a larger number of groups.

8.2 SAMPLING EVENT AND DATA SETS

Parsing over every possible data and event set is the most time consuming portion of our algorithm. To make the procedure somewhat plausible for for larger data sets, sampling, or some other form of approximation must be performed. The main technique we are currently assessing to solve this problem only finds a set with a high MI, not the set with the best MI. This means that a large sample of sets of the appropriate size could be taken and tested for high values of MI. If a set was was found that set could be used for the decomposition. If a set was not found, another sample could be taken and tested. This would repeat until an adequate set was found. This technique relies mainly on the fact that MI is the amount of "connectedness" between groups. Finding a groups with large MI values should produce acceptable results, in theory.

CHAPTER 9

CONCLUSIONS

WACMI is an effective measure to decompose SI BNs such that the resulting network requires little or no communication. Greater than 90% of the networks decomposed using WACMI performed with an accuracy over 80% while requiring no communication. Using our algorithm, effective BN decomposition can be achieved that does not suffer from the inflexibility of the MSBN model. These benefits come as a trade-off with accuracy as our decomposition is only an approximation of the original BN.

In this paper we showed a new measure (WACMI) that can be used to determine the effect of one group of variables on another in a BN. We also demonstrated an algorithm that uses WACMI to effectively decompose an SI BN into two subproblems that require little or no communication. We have also demonstrated various changes that can be made to increase the speed and usefulness of the WACMI decomposition algorithm.

This work adds yet another tool in the ever expanding arsenal of techniques already in use by scientists studying distributed sensor interpretation problems. Our algorithm has benefits over approximation and exact inference methods. It has a flexibility that most decomposition methods do not possess, while still representing a good approximation of

the overall SI BN. Despite the positive aspects of the network, it still contains the main flaw of most DSI systems, run time.

The algorithm is O(n!) where n is the number of data and events combined. However, unlike many of the other systems it does show potential for improvement. The calculation for WACMI and the parsing of sets can potentially be sampled. Another potential issue is the system has yet to be tested on larger networks (> 7-14) and may be hindered in the future by run time and communication needs. Either way *weighted average conditional mutual information* is a useful formula for determining subnet interaction.

REFERENCES

[1] N. Carver and V. Lesser, "Domain monotonicity and the performance of local solutions strategies for cdps-based distributed sensor interpretation," *International Journal of Autonomous Agents and Multi-Agent Systems, vol.* 6, pp. 36-76, 2003.

[2] J. Shen and V. Lesser, "Communication management using abstraction in distributed Bayesian networks," *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems*, ACM Press, pp. 622-629. 2006.

[3] J. Cheng, D. Bell and W. Liu, "Learning Bayesian Networks From Data: an Efficient Approach Based on Information Theory," *Proceeding of the sixth ACM International Conference on Information and Knowledge Management*, 1997.

[4] N. Carver & B. Haan "An empirical analysis of solution quality in distributed bayesian networks from marginalization" Technical Report, Computer Science Department, Southern Illinois University, 2006.

[5] Y. Xiang, D. Pool, & M. Beddoes, "Multiply Sectioned Bayesian Networks and Junction Forests for Large Knowledge-Based Systems," Computer Intelligence Vol.9, No.2, pp. 171-220, 1993.

[6] N. Carver, "A New Framework for Inference in Distributed Bayesian Networks for Multi-Agent Sensor Interpretation," Proceedings of the International Conference on Computers and Their Applications (CATA 2007), March 2007.

[7] Y. Xiang, & V. Lesser, "Justifying Multiply Sectioned Bayesian Networks", *Proceedings of the Fourth International Conference on Multi-Agent Systems* (ICMAS-2000), Boston, MA, July 2000.

VITA

Graduate School
Southern Illinois University

Benjamin Haan Date of Birth: June 11, 1983

108 5th Street, Cordova, Illinois 61242

Illinois State University
Bachelor of Science, Computer Science, May 2005

Special Honors and Awards:
Honors Program Scholar
Graduated Cum Laude
University Honors Scholor
Departmental Honors in School of Information Technology

Thesis Title:

Decomposing Bayesian Network Representations of Distributed Sensor Interpretation Problems Using Weighted Average Conditional Mutual Information

Major Professor: Dr. Norman F. Carver III