

An Investigation of the Cost and Accuracy Tradeoffs of Supplanting AFDs with Bayes Network  
in Query Processing in the Presence of Incompleteness in Autonomous Databases

by

Rohit Raghunathan

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

Approved August 2011 by the  
Graduate Supervisory Committee:

Subbarao Kambhampati, Chair  
Joohyung Lee  
Huan Liu

ARIZONA STATE UNIVERSITY

December 2011

## ABSTRACT

As the information available to lay users through autonomous data sources continues to increase, mediators become important to ensure that the wealth of information available is tapped effectively. A key challenge that these information mediators need to handle is the varying levels of incompleteness in the underlying databases in terms of missing attribute values. Existing approaches such as Query Processing over Incomplete Autonomous Databases (QPIAD) aim to mine and use Approximate Functional Dependencies (AFDs) to predict and retrieve relevant incomplete tuples. These approaches make independence assumptions about missing values—which critically hobbles their performance when there are tuples containing missing values for multiple correlated attributes. In this thesis, I present a principled probabilistic alternative that views an incomplete tuple as defining a distribution over the complete tuples that it stands for. I learn this distribution in terms of Bayes networks. My approach involves mining/learning Bayes networks from a sample of the database, and using it to do both imputation (predict a missing value) and query rewriting (retrieve relevant results with incompleteness on the query-constrained attributes, when the data sources are autonomous). I present empirical studies to demonstrate that (i) at higher levels of incompleteness, when multiple attribute values are missing, Bayes networks do provide a significantly higher classification accuracy and (ii) the relevant possible answers retrieved by the queries reformulated using Bayes networks provide higher precision and recall than AFDs while keeping query processing costs manageable.

*To Amma and Appa*

## ACKNOWLEDGMENTS

Dr. Subbarao Kambhampati is the strongest reason behind my successful graduate experience. I am extremely grateful to him for showing immense patience and believing in my abilities during the phase of research problem identification. His wisdom, constructive criticism and honest feedback have been invaluable and will remain as an example and inspiration for the rest of my life.

My association with Dr. Liu dates back to my first semester here at ASU. He was instrumental in my acclimatization to the new environment and also has been a great source of inspiration throughout my graduate study.

Dr. Joohyung Lee's valuable inputs during the discussion session held regarding my thesis proposal and its subsequent defense were very helpful.

I would like to thank Mike and all the staff members at Hispanic Research Center for their support over the past two years.

I would like to extend my gratitude to members of the Yochan Research Lab- Sushovan, Yuheng, Raju, Manish, J., Will, Kartik, Tuan and friends- Dananjayan, Siva, Rajagopalan, Ganesh, Girish, Bipin, Hari, Harish, Siddharth.

My special thanks goes out to my brother, Rohan and sister-in-law, Ileana who have constantly supported me over the last two years.

One of Victor Hugo's well known quote reads as "Change your opinions, keep to your principles; change your leaves, keep intact your roots." I am fortunate to have as my backbone, my entire family. I will always be indebted to my parents, grandparents and the rest of the family for their support throughout my life, past, present and beyond.

## TABLE OF CONTENTS

	Page
LIST OF TABLES . . . . .	vi
LIST OF FIGURES . . . . .	vii
CHAPTER	
1 INTRODUCTION . . . . .	1
1.1 Background on Incompleteness in Autonomous Databases . . . . .	1
1.1.1 Overview of QPIAD . . . . .	2
1.1.2 Shortcomings of AFD-based Imputation and Query Rewriting . . . . .	3
1.1.3 Bayes Networks . . . . .	4
1.2 Overview of the Thesis . . . . .	5
1.3 Outline . . . . .	6
2 LEARNING BAYES NETWORK MODELS AND IMPUTATION . . . . .	7
2.1 Learning Bayes Network Models for Autonomous Databases . . . . .	7
2.1.1 Structure and Parameter Learning . . . . .	8
2.1.2 Inference in Bayes networks . . . . .	9
2.2 Dealing with Incompleteness in Autonomous Databases using Imputation . . . . .	9
2.2.1 Imputing Single Missing Values . . . . .	10
2.2.2 Imputing Multiple Missing Values . . . . .	10
2.2.3 Prediction Accuracy with Increase in Incompleteness in Test Data . . . . .	11
2.2.4 Time Taken for Imputation . . . . .	12
2.3 Summary . . . . .	15
3 HANDLING INCOMPLETENESS IN AUTONOMOUS DATABASES USING QUERY REWRITING . . . . .	16
3.1 Generating Rewritten Queries using Bayes Networks . . . . .	17
3.1.1 Generating Rewritten Queries using <i>BN-Beam</i> . . . . .	19
3.1.2 Handling Multi-attribute Queries . . . . .	21
3.2 Empirical Evaluation of Query Rewriting . . . . .	22
3.2.1 Comparison of Rewritten Queries Generated by AFDs and BN-All-MB . . . . .	23
3.2.2 Comparison of Rewritten Queries Generated by BN-All-MB and BN-Beam . . . . .	23
3.2.3 Comparison of Multi-attribute Queries . . . . .	24
3.3 Summary . . . . .	30
4 CONCLUSION & FUTURE WORK . . . . .	33

Chapter	Page
4.1 Conclusion . . . . .	33
4.2 Future Work . . . . .	33
4.2.1 Generating Rewritten Queries in the Presence of Limited Query Patterns	33
4.2.2 Reducing CPU Time in Generating Rewritten Queries using BN-Beam . .	34
4.2.3 Redundancy Metrics to Improve Recall . . . . .	34
REFERENCES . . . . .	35

## LIST OF TABLES

Table	Page
1.1 A Fragment of a car database. . . . .	4
2.1 Domain size of attributes in Car database . . . . .	7
2.2 Domain size of attributes in Adult database . . . . .	7
2.3 Time taken for Imputation by AFDs, BN-Gibbs and BN-Exact . . . . .	15
3.1 A Fragment of a car database . . . . .	16
4.1 A Fragment of a car database . . . . .	34

## LIST OF FIGURES

Figure	Page
2.1 A Bayesian network learnt from a sample of data extracted from <i>Cars.com</i> . . . . .	8
2.2 A Bayesian network learnt for Adult database . . . . .	8
2.3 Single Attribute Prediction Accuracy (Cars) . . . . .	10
2.4 Multiple Attribute Prediction Accuracy (Cars) . . . . .	12
2.5 Prediction Accuracy of <i>Model</i> with increase in percentage of incompleteness in test data . . . . .	13
2.6 Prediction Accuracy of <i>Year-Body</i> with increase in percentage of incompleteness in test data . . . . .	13
2.7 Prediction Accuracy of <i>Education</i> with increase in percentage of incompleteness in test data . . . . .	14
2.8 Prediction Accuracy of <i>Race-Occupation</i> with increase in percentage of incompleteness in test data . . . . .	14
3.1 Precision-Recall curve for query $\sigma_{Make}$ . . . . .	24
3.2 Precision-recall curve for query $\sigma_{Body}$ averaged over 3 queries . . . . .	25
3.3 Precision-Recall curve for query $\sigma_{Relationship=Not-in-family}$ . . . . .	25
3.4 Change in recall for different values of $\alpha$ in F-measure metric for top-10 rewritten queries for $\sigma_{Year=2002}$ . . . . .	26
3.5 Change in precision for different values of $\alpha$ in F-measure metric for top-10 rewritten queries for $\sigma_{Year=2002}$ . . . . .	26
3.6 Precision-recall curve for the results returned by top-10 rewritten queries for the query $\sigma_{Make=bmw \wedge Mileage=15000}$ . . . . .	28
3.7 Change in recall as the number of queries sent to the autonomous database increases for the query $\sigma_{Make=bmw \wedge Mileage=15000}$ . . . . .	28
3.8 Precision-recall curve for the results returned by top-10 rewritten queries for the query $\sigma_{Education=HS-grad \wedge Relationship=Husband}$ . . . . .	29
3.9 Change in recall as the number of queries sent to the autonomous database increases for the query $\sigma_{Education=HS-grad \wedge Relationship=Husband}$ . . . . .	29
3.10 Precision-recall curve for the results returned by top-10 rewritten queries for the query $\sigma_{Make=kia \wedge Year=2004}$ . . . . .	30
3.11 Change in recall as the number of queries sent to the autonomous database increases for the query $\sigma_{Make=kia \wedge Year=2004}$ . . . . .	30



Figure	Page
3.12 Precision-recall curve for the results returned by top-10 rewritten queries for the query $\sigma_{WorkClass=private \wedge Relationship=own-child}$ . . . . .	31
3.13 Change in recall as the number of queries sent to the autonomous database increases for the query $\sigma_{WorkClass=private \wedge Relationship=own-child}$ . . . . .	31

### INTRODUCTION

As the popularity of the World Wide Web continues to increase, lay users have access to more and more information in autonomous databases. Incompleteness in these autonomous sources is extremely commonplace. Such incompleteness mainly arises due to the way in which these databases are populated- through (inaccurate) automatic extraction or by lay users. Dealing with incompleteness in the databases requires tools for dealing with uncertainty. Previous attempts at dealing with this uncertainty by systems like QPIAD [15] have mainly focused on using rule-based approaches, popularly known in the database community as Approximate Functional Dependencies (AFDs). The appeal of AFDs is due to the ease of specifying the dependencies, learning and reasoning with uncertainty. However, uncertain reasoning using AFDs adopts the certainty factors model, which assumes that the principles of locality and detachment [13] hold. But, these principles do not hold for uncertain reasoning and can lead to erroneous reasoning. As the levels of incompleteness in the information sources increases, the need for more scalable and accurate reasoning becomes paramount. Full probabilistic reasoning avoids the traps of AFDs. Graphical models are an efficient way of doing full probabilistic reasoning. Bayesian network (Bayes net) is such a model, where direct dependencies between the variables in a problem are modeled as a directed acyclic graph, and the indirect dependencies can be inferred. As desired, Bayes nets can model both causal and diagnostic dependencies. Using Bayes nets for uncertain reasoning has largely replaced rule-based approaches in Artificial Intelligence. However, learning and inference on Bayes nets can be computationally expensive which might inhibit their applications to handling incompleteness in autonomous data sources. In this thesis, we consider if these costs can be handled without compromising on the improved accuracy offered by Bayes nets, in the context of incompleteness in the autonomous databases.

#### *1.1 Background on Incompleteness in Autonomous Databases*

Increasingly many of the autonomous web databases are being populated by automated techniques or by lay users, with very little curation. For example, databases like auto-trader.com are populated using automated extraction techniques by crawling the text classifieds and by car owners entering data through forms. Scientific databases such as CbioC [3], also use similar techniques for populating the database. However, Gupta and Sarawagi [5] have shown that these techniques are error prone and lead to incompleteness in the database in

the sense that many of the attributes have missing values. Wolf et al [15] report that 99% of the 35,000 tuples extracted from Cars Direct were incomplete. When the mediator has privileges to modify the data sources, the missing values in these data sources can be completed using "imputation", which attempts to fill in the missing values with the most likely value. As the levels of incompleteness in these data sources increases, it is not uncommon to come across tuples with multiple missing values. Effectively finding the most likely completions for these multiple missing values would require capturing the dependencies between them. A second challenge arises when the underlying data sources are autonomous, i.e., access to these databases are through forms, the mediator cannot complete the missing values with the most likely values. Therefore, mediators need to generate and issue a set of reformulated queries, in order to retrieve the relevant answers with missing values. Efficiency considerations dictate that the number of reformulations be kept low. In such scenarios, it becomes very important for mediators to send queries that not only retrieve results with a large fraction of relevant results (precision), but also a large number of relevant results (recall).

### 1.1.1 Overview of QPIAD

The QPIAD system [15] addresses the challenges in retrieving relevant incomplete answers by learning the correlations between the attributes in the database as AFDs and the value distributions as Naive Bayesian Classifiers.

Given a Relation  $R$ , a subset  $X$  of its attributes, and a single attribute  $A$  of  $R$ , an approximate functional dependency (AFD) holds on a Relation  $R$ , between  $X$  and  $A$ , denoted by,  $X \rightsquigarrow A$ , if the corresponding functional dependency  $X \rightarrow A$  holds on all but a small fraction of tuples of  $R$ .

To illustrate how QPIAD works consider the the query  $Q : Body=SUV$  issued to Table 1.1. Traditional query processors will only retrieve tuples  $t_7$  and  $t_9$ . However, the entities represented by tuples  $t_8$  and  $t_{10}$  are also likely to be relevant. The QPIAD system's aim is to retrieve tuples  $t_8$  and  $t_{10}$ , in addition to  $t_7$  and  $t_9$ . In order to retrieve tuples  $t_8$  and  $t_{10}$  it uses AFDs mined from a sample of the database. For example, an AFD  $Model \rightsquigarrow Body$  may be mined for the fragment of the cars database shown in Table 1.1. This indicates that the value of a car's *Model* attribute *often* (but not always) determines the value of its *Body* attribute. These rules are used to retrieve relevant incomplete answers.

When the mediators have access privileges to modify the database, AFDs are used

along with Naive Bayes Classifiers to fill in the missing values as a simple classification task and then traditional query processing will suffice to retrieve relevant answers with missing values. However, in more realistic scenarios, when such privileges are not provided, mediators generate a set of rewritten queries and send to the database, in addition to the original user query. According to the AFD mentioned above and tuple  $t_7$  retrieved by traditional query processors, a rewritten query  $Q'_1 : \sigma_{Model=Santa}$  may be generated to retrieve  $t_8$ . Similarly  $Q'_2 : \sigma_{Model=MDX}$  may be generated which will retrieve  $t_{10}$ .

Multiple rules can be mined for each attribute, for example, the mileage and year of a car might determine the body style of the car. So a rule  $\{Year, Mileage\} \rightsquigarrow \{Body\}$  could be mined. Each rule has a confidence associated with it, which specifies how accurate the determining set of an attribute's AFD is in predicting it. The current QPIAD system uses only the highest confidence AFD<sup>1</sup> of each attribute for imputation and query rewriting. In addition, it only aims to retrieve relevant incomplete answers with at most one missing value on query-constrained attributes.

### 1.1.2 Shortcomings of AFD-based Imputation and Query Rewriting

AFDs are rule-based methods for dealing with uncertainty. AFDs adopt the certainty factors model which makes two strong assumptions:

1. **Principle of Locality:** Whenever there is a rule  $A \rightarrow B$ , given evidence of A, we can conclude B, regardless of the other rules and evidences.
2. **Principle of Detachment:** Whenever a proposition B is found to be true, the truth of B can be used regardless of how it was found to be true.

However, these two assumptions do not hold in the presence of uncertainty. When propagating beliefs, not only is it important to consider *all* the evidences but also their sources. Therefore, using AFDs for reasoning with uncertainty can lead to cyclic reasoning and fail to capture the correlations between multiple missing values. In addition to these shortcomings, the beliefs are represented using a Naive-Bayesian Classifier, which makes strong conditional independence assumptions, often leading to inaccurate values.

To illustrate the shortcomings of AFDs, consider a query  $Q : \sigma_{Model=A8 \wedge Year=2005}$  issued to the fragment of the car database shown in Table 1.1. When the mediator has modification privileges, the missing values for attributes *Model* and *Year* can be completed with the most

<sup>1</sup>The actual implementation of QPIAD uses a variant to the highest confidence AFD for some of the attributes. For details we refer the reader to [15]

ID	Make	Model	Year	Body	Mileage
1	Audi	null	null	Sedan	20000
2	Audi	A8	null	Sedan	15000
3	BMW	745	2002	Sedan	40000
4	Audi	null	2005	Sedan	20000
5	Audi	A8	2005	Sedan	20000
6	BMW	645	1999	Convrt	null
7	Hyundai	Santa	1990	SUV	45000
8	Hyundai	Santa	1993	null	40000
9	Acura	MDX	1990	SUV	30000
10	Acura	MDX	1990	null	12000

Table 1.1: A Fragment of a car database.

likely values, before returning the answer set. Using AFDs to predict the missing values in tuple  $t_1$ , ignores the correlation between the *Model* and *Year*; predicting them independently. Substituting the value for missing attribute *Year* in tuple  $t_2$  using just the highest confidence rule as is done in QPIAD [15], often leads to inaccurate propagation of beliefs as the other rules are ignored. When the mediator does not have privileges to modify the database, a set of rewritten queries are generated and issued to the database to retrieve the relevant uncertain answers. Issuing  $Q$  to the database fragment in Table 1.1 retrieves  $t_5$ . The rewritten queries generated by methods discussed in QPIAD [15] retrieve tuples  $t_2$  and  $t_4$ . However, it does not retrieve tuple  $t_1$ , but it is highly possible that the entity represented by it is relevant to the user's query.

### 1.1.3 Bayes Networks

A Bayes network [8] is a graphical representation of the probabilistic dependencies between the variables in a domain. The generative model of a relational database can be represented using a Bayes network, where each node in the network represents an attribute in the database. The edges between the nodes represent direct probabilistic dependencies between the attributes. The strength of these probabilistic dependencies are modeled by associating a conditional probability distribution (CPD) with each node, which represents the conditional probability of a variable, given as evidence each combination of values of its immediate parents. A Bayes network is a compact representation of the full joint probability distribution of the nodes in the network. The full joint distribution can be constructed from the CPDs in the Bayes network. Given the full joint distribution, any probabilistic query can be answered. In particular, the probability of any set of hypotheses can be computed, given any set of observations, by conditioning and marginalizing over the joint distribution. Since the semantics of Bayes networks are

in terms of the full joint probability distribution, inference using them considers the influence of all variables in the network. Therefore, Bayes nets, unlike AFDs, do not make the Locality and Detachment assumptions.

## 1.2 Overview of the Thesis

Given the advantages of Bayes networks over AFDs, we investigate if replacing AFDs with a Bayes network in QPIAD system [15], provides higher accuracy and while keeping the costs manageable. Learning and inference with Bayes networks are computationally harder than AFDs. Therefore, the challenges involved in replacing AFDs with Bayes networks include learning and using them to do both imputation and query rewriting by keeping costs manageable. We use BANJO software package [1] to learn the topology of the Bayes network and use BNT [4] and INFER.NET [10] software packages to do inference on them. Even though learning the topology for the Bayes net from a sample of the database involves searching over the possible topologies, we found that high fidelity Bayes networks could be learnt from a small fraction of the database by keeping costs manageable (in terms of time spent in searching). Inference in Bayes networks is intractable if the network is multiply connected, i.e., there is more than undirected path between any two nodes in the network. We handle this challenge by using approximate inference techniques. Approximate inference techniques are able to retain the accuracy of exact inference techniques and keep the cost of inference manageable. We compare the cost and accuracy of using AFDs and Bayes networks for imputing single and multiple missing values at different levels of incompleteness in test data.

We also develop new techniques for generating rewritten queries using Bayes networks. To illustrate the challenges involved in generating rewritten queries consider the same query  $Q:\sigma_{Model=A8 \wedge Year=2005}$  issued to the database fragment in Table 1.1. our aim is to retrieve tuple  $t_1$  in addition to tuples  $t_2$ ,  $t_4$  and  $t_5$ . The challenges that are involved in generating rewritten queries are:

1. Selecting the attributes on which the new queries will be formulated. Selecting these attributes by searching over all the attributes becomes too expensive as the number of attributes in the database increases.
2. Determining the values to which the attributes in the rewritten query will be constrained to. The size of the domains of attributes in most autonomous databases is often large. Searching over each and every value can be expensive.
3. Most autonomous data sources have a limit on the number of queries to which it will answer.

The rewritten queries that we generate should be able to carefully tradeoff precision with the throughput of the results returned.

We propose techniques to handle these challenges and evaluate them with AFD-based approaches in terms of precision and recall of the results returned.

### *1.3 Outline*

The rest of the thesis is organized as follows— in Chapter 2, we discuss how Bayes network models of autonomous databases can be learnt by keeping costs manageable and compare the prediction accuracy and cost of using Bayes network and AFDs for imputing missing values. Next, in Chapter 3, we discuss how rewritten queries are generated using Bayes networks and compare them with AFD-approaches for single and multi-attribute queries. Finally, we conclude in Chapter 4 by discussing directions for future work.

## LEARNING BAYES NETWORK MODELS AND IMPUTATION

In this chapter we discuss how we learn the topology and parameters of the Bayes network by keeping costs manageable in (Section 2.1). We then compare the prediction accuracy of the learned Bayes network model and AFDs in imputation of single (Section 2.2.1) and multiple (Section 2.2.2) attributes. Next we compare their prediction accuracies at different levels of incompleteness in test data (Section 2.2.3). Finally we report the prediction costs in terms of time taken by the various techniques in (Section 2.2.4).

### 2.1 Learning Bayes Network Models for Autonomous Databases

In this section we discuss how we learn the topology and parameters of the Bayes network. We learn the generative model of two databases- A fragment of 8000 tuples extracted from *Cars.com* [2] and Adult database consisting of 15000 tuples obtained from UCI data repository [12]. Table 2.1 and 2.2 describe the schema and the domain sizes of the attributes in the two databases. The attributes with continuous values are discretized and used as categorical attributes. *Price* and *Mileage* attributes in the cars database are discretized by rounding off to the nearest five thousand. In the adult database attributes *Age* and *Hours Per Week* are discretized to the nearest multiple of five.

Database	Year	Model	Make	Price	Mileage	Body
Cars-8000-20(Mediator)	9	38	6	19	17	5
Cars-8000-100(Complete)	12	41	6	30	20	7

Table 2.1: Domain size of attributes in Car database.

Database	Age	Work Class	Educational	Marital Status	Occupation	Relationship	Race	Sex	Hours Per Week	Native Country
Adult-15000-20(Mediator)	8	7	16	7	14	6	5	2	10	37
Adult-15000-100(Complete)	8	7	16	7	14	6	5	2	10	40

Table 2.2: Domain size of attributes in Adult database.



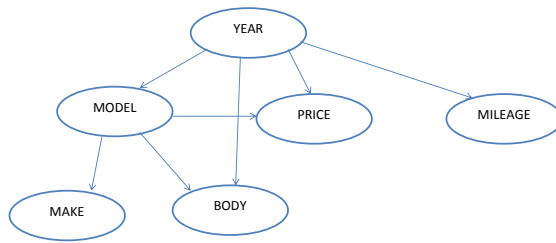


Figure 2.1: A Bayesian network learnt from a sample of data extracted from *Cars.com*

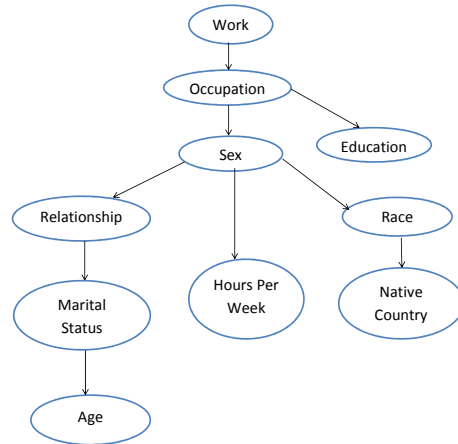


Figure 2.2: A Bayesian network learnt for Adult database

### 2.1.1 Structure and Parameter Learning

The structure of the Bayes network is learned from a complete sample of the autonomous database. We use the BANJO package [1] as a black box for learning the structure of the Bayes network. To keep the learning costs manageable we constrain nodes to have at most two parents. In cases where there are more than two attributes directly correlated to an attribute, these attributes can be modeled as children. There is no limit on the number of children a node can have. Figure 2.1.1 shows the structure of a Bayes network learned for the Cars database and Figure 2.1.1 for the Adult database. We used samples of sizes varying from 5-20% of the database and found that the structure of the highest scoring network remained the same. We also experimented with different time limits for the search, ranging from 5-30 minutes. We did not see any change in the structure of the highest confidence network.

### 2.1.2 Inference in Bayes networks

Bayes network inference is used in both imputation and query rewriting tasks. Imputation involves substituting the missing values with the most likely values, which involves inference. Exact inference in Bayes networks is NP-hard [13] if the network is multiply connected. Therefore, to keep query processing costs manageable we use approximate inference techniques. In our experiments described in section 2.2, we found that using approximate inference techniques retains the accuracy edge of exact inference techniques, while keeping the prediction costs manageable. We use the BNT package [4] for doing inference on the Bayes Network for the imputation task. We experimented with various exact inference engines that BNT offers and found the junction-tree engine to be the fastest. While querying multiple variables, junction tree inference engine can be used only when all the variables being queried form a clique. When this is not the case, we use the variable elimination inference engine.

### 2.2 Dealing with Incompleteness in Autonomous Databases using Imputation

In this section we compare the prediction accuracy and cost of Bayes networks versus AFDs for imputing single and multiple missing values when there is incompleteness in test data. When the mediator has privileges to modify the underlying Autonomous database, the missing values can be substituted with the most probable value. Imputation using Bayesian networks first computes the posterior of the attribute that is to be predicted given the values present in the tuple and completes the missing value with the most likely value given the evidence. When predicting multiple missing values, the joint posterior distribution over the missing attributes are computed and the values with the highest probability are used for substituting the missing values. Computing the joint probability over multiple missing values captures the correlations between the missing values, which gives Bayes networks a clear edge over AFDs. In contrast, imputation using AFDs uses the AFD with the highest confidence for each attribute for prediction. If an attribute in the determining set of an AFD is missing, then that attribute is first predicted using other AFDs (chaining), before the original attribute can be predicted. The most likely value for each attribute is used for completing the missing value. When multiple missing values need to be predicted, each value is predicted independently.

We use the Cars and Adult databases described in the previous section. We compare AFD approach used in QPIAD which uses Naive Bayesian Classifiers to represent value distributions with exact and approximate inference in Bayes networks. We call exact inference in

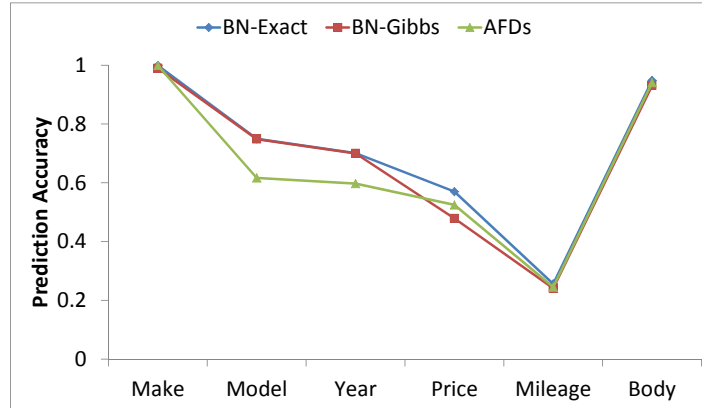


Figure 2.3: Single Attribute Prediction Accuracy (Cars)

Bayes network as BN-Exact. We use Gibbs sampling as the approximate inference technique, which we call BN-Gibbs. For BN-Gibbs, the probabilities are computed using 250 samples.

### 2.2.1 Imputing Single Missing Values

Our experiments show that prediction accuracy using Bayes nets is higher than AFDs for attributes which have multiple high confidence rules. Approaches for combining multiple rules for classification have been shown to be ineffective by Khatri [7]. Since there is no straightforward way for propagating beliefs using multiple AFDs, only the AFD with the highest confidence is used for propagating beliefs. This method, however, fails to take advantage of additional information that the other rules provide. Bayes networks, on the other hand, systematically combine evidences from multiple sources. Figure 2.3 shows the prediction accuracy in the presence of a single missing value for each attribute in the Cars database. We notice that there is a significant difference in prediction accuracies of attributes *Model* and *Year*. There are multiple rules that are mined for these two attributes but using just the rule with highest confidence, ignores the influence of the other available evidence, which affects the prediction accuracy.

### 2.2.2 Imputing Multiple Missing Values

In most real-world scenarios, however, the number of missing values per tuple is likely to be more than one. The advantage of using a more general model like Bayes networks becomes even more apparent in these cases. Firstly, AFDs cannot be used to impute all combinations of missing values, this is because when the determining set of an AFD contains a missing attribute, then the value needs to be first predicted using a different AFD by chaining. While

chaining, if we come across an AFD containing the original attribute to be predicted in its determining set, then predicting the missing value becomes impossible. When the missing values are highly correlated, AFDs often get into such cyclic dependencies. In Figure 2.4 we can see that the attribute pairs *Year-Mileage*, *Body-Model* and *Make-Model* cannot be predicted by AFDs. As the number of missing values increases, the number of combinations of missing values that can be predicted reduces. In our experiments with the Cars database, when predicting three missing values, only 9 out of the 20 possible combinations of missing values could be predicted.

On the other hand, Bayes networks can predict the missing values regardless of the number and combination of values missing in a tuple. Bayes networks also predict missing values in extreme cases where every value in a tuple is missing. Secondly, while predicting the missing values, Bayes nets compute the joint probability distribution over the missing attributes which allows them to capture the correlations between the attributes. In contrast, for whereas prediction using AFDs, which use a Naïve Bayes Classifier to represent the value distributions, predict each of the missing attributes independently, ignoring the interactions between them. The attribute pair *Year-Model* in Figure 2.4 shows that the prediction accuracy is significantly higher when correlations between the missing attributes are captured. We also observe that in some cases, when the missing values are D-separated [14] given the values for other attributes in the tuple, the performance of AFDs and Bayes networks is comparable. In Figure 2.4, we can see that the prediction accuracy for *Mileage-Make* and *Mileage-Body* are comparable for all the techniques since attributes are D-separated given the other evidence in the tuple. However, the number of attributes that are D-separated is likely to decrease with increase in incompleteness in the databases.

### 2.2.3 Prediction Accuracy with Increase in Incompleteness in Test Data

As the incompleteness in the database increases, not only does the number of values that need to be predicted increase, but also the evidence for predicting these missing values reduces. Therefore, it is important for the classifier to be robust with the increase in incompleteness in the underlying autonomous databases. We compared the performance of Bayes nets and AFDs as the incompleteness in the autonomous databases increase. We see that the prediction accuracy of AFDs drops faster with increase in incompleteness. As mentioned earlier, This is because when the determining set of an AFD contains an attribute whose value is missing in the current tuple, then that value needs to be first predicted using a different AFD (chaining),

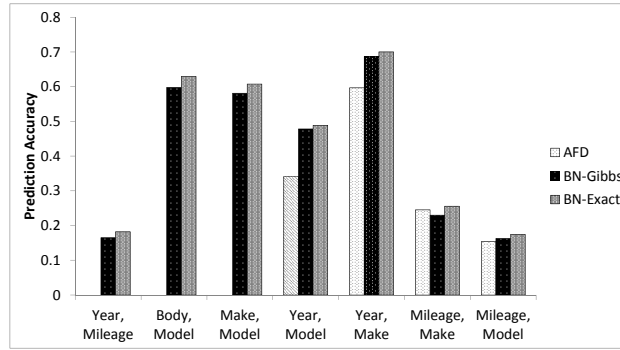


Figure 2.4: Multiple Attribute Prediction Accuracy (Cars)

before the original missing attribute can be predicted. In general, when an attribute has multiple AFDs (rules), propagating beliefs using just one rule and ignoring the others, often violates the principles of detachment and locality [6]. As the levels of incompleteness of the database increases, so does the number of times chaining is done, which affects the prediction accuracy. Another problem that arises while propagating beliefs through chaining is that we often come across an AFD containing the original attribute to be predicted in its determining set, causing a cyclic dependency which renders prediction impossible. Therefore, as the number of missing values increases, so does the possibility that the AFD might get into a cyclic dependency. On the other hand, Bayesian networks, being a generative model, can infer the values of any set of attributes given the evidence of any other set. Therefore, as the incompleteness of the database increases, the prediction accuracy of Bayes Networks will be significantly higher than that of AFDs. Our empirical results show in Figure 2.5 and 2.7 show the prediction accuracy of AFDs and Bayes nets when a single attribute needs to be predicted in the Cars and Adult databases respectively. Figures 2.6 and 2.8 show the prediction accuracy for predicting missing values on two attributes on the Cars and Adult databases respectively. We see that both Bayes net approaches have a higher prediction accuracy than AFDs at all levels of incompleteness.

#### 2.2.4 Time Taken for Imputation

We now compare the time taken for imputing the missing values using AFDs, Exact Inference (Junction tree) and Gibbs Sampling (250 samples) are compared as the number of missing values in the autonomous database increases. Table 2.3 reports the time taken to impute a Cars database with 5479 tuples. We note that while imputation is most accurate when using Exact Inference, the preferred method for most applications is Gibbs sampling, as it's accuracy is not

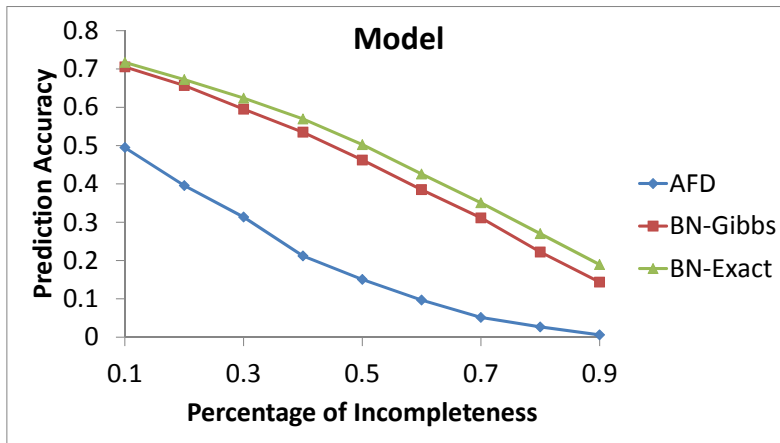


Figure 2.5: Prediction Accuracy of *Model* with increase in percentage of incompleteness in test data

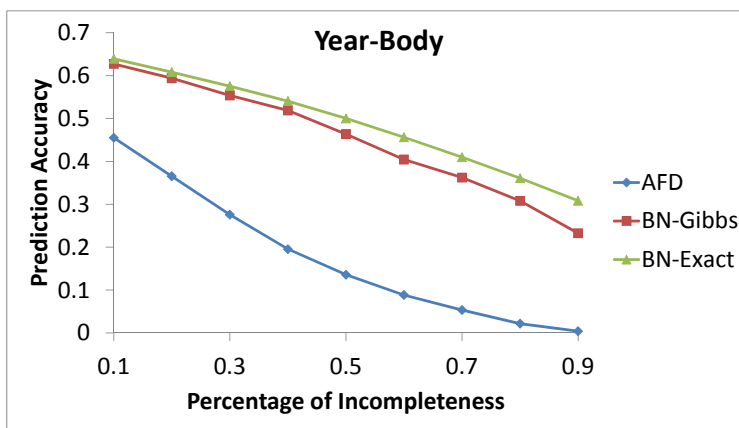


Figure 2.6: Prediction Accuracy of *Year-Body* with increase in percentage of incompleteness in test data

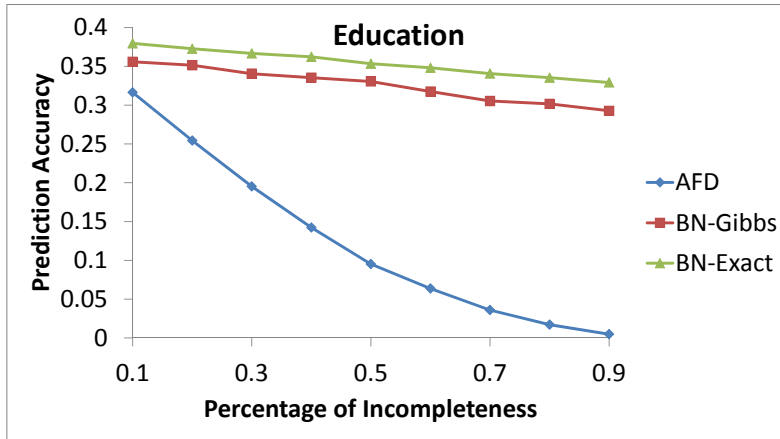


Figure 2.7: Prediction Accuracy of *Education* with increase in percentage of incompleteness in test data

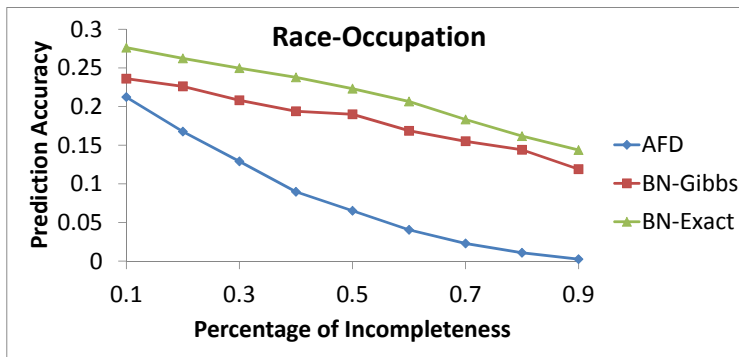


Figure 2.8: Prediction Accuracy of *Race-Occupation* with increase in percentage of incompleteness in test data

Percentage of Incompleteness	Time Taken for AFD(Sec.)	Time Taken by BN-Gibbs (Sec.)	Time Taken by BN-Exact (Sec.)
0%	0.271	44.46	16.23
10%	0.267	47.15	44.88
20%	0.205	52.02	82.52
30%	0.232	54.86	128.26
40%	0.231	56.19	182.33
50%	0.234	58.12	248.75
60%	0.232	60.09	323.78
70%	0.235	61.52	402.13
80%	0.262	63.69	490.31
90%	0.219	66.19	609.65

Table 2.3: Time taken for predicting 5479 tuples by AFDs, BN-Gibbs (250 Samples) and BN-Exact in seconds.

very far off from exact inference (see Figures 2.5 – 2.8), while keeping the cost of inference more manageable.

### 2.3 Summary

In this chapter we discussed how a Bayes network model of an autonomous database can be learnt from a sample of the database by keeping costs manageable. In particular, learning and using Bayes network models for imputation can be divided into 2 parts– (i) *Topology and Parameter learning*: High-fidelity topologies can be learnt from a small fraction of the database and with manageable cost by bounding the number of parents per node and (iii) *Inference*: Inference in Bayes networks is intractable. But approximate inference techniques retain the accuracy edge of exact methods while keeping costs manageable.

We then showed that Bayes networks have a higher prediction accuracy than AFDs because they systematically combine evidences from multiple sources whereas AFDs fail to do so; winding up using only the highest confidence AFD. This shortcoming becomes apparent as the levels of incompleteness in test data increases, and when the missing values are highly correlated.



## HANDLING INCOMPLETENESS IN AUTONOMOUS DATABASES USING QUERY REWRITING

In this chapter we describe BN-All-MB, a technique for retrieving relevant incomplete results from autonomous databases using Bayes networks, when the query processors are allowed read-only access. We compare BN-All-MB and AFD approach for handling single-attribute queries in Section 3.2.1. We then discuss the need for techniques that can reformulate queries when the autonomous data sources have limits on the number of queries that they respond to. Next, in Section 3.1.1 we propose BN-Beam, our technique for handling this challenge and evaluate it in Section 3.2.2. Finally in Section 3.2.3 we compare BN-Beam and AFD-based approaches for handling multi-attribute queries when there are limits on the number of queries that can be issued to the autonomous database.

In information integration scenarios when the underlying data sources are autonomous (i.e., the query processor is allowed read-only access without the capability to write or modify the underlying database), missing values cannot be completed using a classification task such as the imputation method discussed in the previous chapter. Our goal is to retrieve *all* relevant answers to the user's query, including tuples which are relevant, but have missing values on the attributes constrained in the user's query. Since query processors are allowed read-only access to these databases, the only way to retrieve the relevant answers with missing values on query-constrained attributes is by generating and sending a set of reformulated queries that constrain other relevant attributes.

ID	Make	Model	Year	Body	Mileage
1	Audi	A8	2005	Sedan	20000
2	Audi	A8	2005	null	15000
3	Acura	tl	2003	Sedan	null
4	BMW	745	2002	Sedan	40000
5	null	745	2002	Sedan	null
6	null	645	1999	Convrt	null
7	null	645	1999	Coupe	null
8	null	645	1999	Convrt	null
9	BMW	645	1999	Coupe	40000
10	BMW	645	1999	Convrt	40000

Table 3.1: A Fragment of a car database

### 3.1 Generating Rewritten Queries using Bayes Networks

We will use a fragment of the Car database shown in Table 3.1 to explain our approach. Notice that tuples  $t_2, t_3$  have one missing (null) value and tuples  $t_5, t_6, t_7, t_8$  have two missing values. To illustrate query rewriting when a single attribute is constrained in the query, consider a query( $Q$ )  $\sigma_{Body=Sedan}$ . First, the query  $Q$  is issued to the autonomous database and all the certain answers which correspond to tuples  $t_1, t_3, t_4$  and  $t_5$  in the Table 3.1 are retrieved. This set of certain answers forms the *base result set*. However, tuple  $t_2$ , which has a missing value for *Body* (possibly due to incomplete extraction or entry error), is likely to be relevant since the value for *Body* should have been *Sedan* had it not been missing. In order to determine the attributes and their values on which the rewritten queries need to be generated, we use the Bayesian network learnt from the sample of the autonomous database.

Using the same example, we now illustrate how rewritten queries are generated. First, the set of certain answers which form the base result set are retrieved and returned to the user. The attributes on which the new queries are reformulated consist of all attributes in the *markov blanket* of the original query-constrained attribute. The markov blanket of a node in a Bayesian network consists of its parent nodes, children nodes and children's other parent nodes. We consider all attributes in the markov blanket while reformulating queries because given the values of these attributes, the original query-constrained attribute is dependent on no other attribute in the Bayesian network. From the learnt Bayesian network shown in Figure 2.1.1, the markov blanket of the attribute *Body* consists of attributes  $\{Year, Model\}$ . The value that each of the attributes in the rewritten query can be constrained to is limited to the distinct value combinations for each attribute in the base result set. This is because, the values that the other attributes take are highly likely to be present in relevant incomplete tuples. This tremendously reduces the search effort required in generating rewritten queries, without affecting the recall too much. However, at higher levels of incompleteness, this might have a notable impact on recall, in which case, we would search over the values in the entire domain of each attribute. Continuing our example, when the query  $Q$  is sent to the database fragment shown in Table 3.1, tuples  $t_1, t_3, t_4$  and  $t_5$  are retrieved. The values over which the search is performed for *Model* is  $\{A8, tl, 745\}$  and for *Year* is  $\{2002, 2003, 2005\}$ .

Some of the rewritten queries that can be generated by this process are  $Q'_1 : \sigma_{Model=A8 \wedge Year=2005}$ ,  $Q'_2 : \sigma_{Model=tl \wedge Year=2003}$  and  $Q'_3 : \sigma_{Model=745 \wedge Year=2002}$ . Each of these

queries differ in the number of results that they retrieve and the fraction of retrieved results that are relevant. An important issue here is to decide which of these queries should be issued to the autonomous database and in which order. If we are allowed to send as many queries as we want, ordering the queries in terms of their *expected precision* would obviate the need for ranking the relevant possible results once they are retrieved. This is because the probability that the missing value in a tuple is exactly the values the user is looking for is the same as the expected precision of the query that retrieves the tuple. However, limits are often imposed on the number of queries that can be issued to the autonomous database. These limits could be due to network or processing resources of the autonomous data sources. Given such limits, the precision of the answers need to be carefully traded off with selectivity (the number of results returned) of the queries. One way to address this challenge is to pick the top- $K$  queries based on the *F-measure* metric [9], as pointed out by Wolf et al [15]. F-measure is defined as the weighted harmonic mean of precision ( $P$ ) and recall ( $R$ ) measures :  $\frac{(1+\alpha)*P*R}{\alpha*P+R}$ . For each rewritten query, the F-measure metric is evaluated in terms of its *expected precision* and *expected recall*. The latter which can be computed from the *expected selectivity*. Expected precision is computed from the Bayesian network and expected selectivity is computed the same way as computed by the QPIAD system [15], by issuing the query to the sample of the autonomous database. For our example, the expected precision of the rewritten query  $\sigma_{Model=A8 \wedge Year=2005}$  can be computed as the  $P(Body=Sedan | Model=A8 \wedge Year=2005)$  which is evaluated by inference on the learned Bayesian network. Expected selectivity is computed as  $SmplSel(Q)*SmplRatio(R)$ , where  $SmplSel(Q)$  is the sample selectivity of the query  $Q$  and  $SmplRatio(R)$  is the ratio of the original database size over the size of the sample. We send queries to the original database and its sample offline and use the cardinalities of the result sets to estimate the ratio.

We now describe the query rewriting algorithm which we call BN-All-MB in detail.

We refer to this technique for generating rewritten queries by constraining all attributes in the markov blanket as *BN-All-MB*. In section 3.2.1 we show that BN-All-MB retrieves uncertain relevant tuples with higher precision than the AFD-approach. However, the issue with constraining all attributes in the markov blanket is that its size could be arbitrarily large. As its size increases, the number of attributes that are constrained in the rewritten queries also increase. This will reduce the throughput of the queries significantly. As we mentioned earlier, in cases where the autonomous database has a limit on the number of queries to which it will respond, we need to carefully trade off precision of the rewritten queries with their throughput. BN-All-MB and AFD approaches decide upfront the attributes to be constrained and search

Let  $R(A_1, A_2, \dots, A_n)$  be a database relation. Suppose  $MB(A_p)$  is the set of attributes in the markov blanket of attribute  $A_p$ . A query  $Q: \sigma_{A_p=v_p}$  is processed as follows

1. Send  $Q$  to the database to retrieve the base result set  $RS(Q)$ . Show  $RS(Q)$ , the set of certain answers, to the user.
2. Generate a set of new queries  $Q'$ , order them, and send the most relevant ones to the database to retrieve the extended result set  $\widehat{RS}(Q)$  as relevant possible answers of  $Q$ . This step contains the following tasks.
  - (a) *Generate Rewritten Queries.* Let  $\pi_{MB(A_p)}(RS(Q))$  be the projection of  $RS(Q)$  onto  $MB(A_p)$ . For each distinct tuple  $t_i$  in  $\pi_{MB(A_p)}(RS(Q))$ , create a selection query  $Q'_i$  in the following way. For each attribute  $A_x$  in  $MB(A_p)$ , create a selection predicate  $A_x=t_i.v_x$ . The selection predicates of  $Q'_i$  consist of the conjunction of all these predicates
  - (b) *Select the Rewritten Queries.* For each rewritten query  $Q'_i$ , compute the estimated precision and estimated recall using the Bayes network as explained earlier. Then order all  $Q'_i$ 's in order of their F-Measure scores and choose the top- $K$  to issue to the database.
  - (c) *Order the Rewritten Queries.* The top- $K$   $Q'_i$ 's are issued to the database in the decreasing order of expected precision.
  - (d) *Retrieve extended result set.* Given the ordered top- $K$  queries  $\{Q'_1, Q'_2, \dots, Q'_K\}$  issue them to the database and retrieve their result sets. The union of result sets  $RS(Q'_1), RS(Q'_2), \dots, RS(Q'_K)$  is the extended result set  $\widehat{RS}(Q)$ .

**Algorithm 1:** Algorithm for BN-All-MB

only over the values to which the attributes will be constrained. Both these techniques try to address this issue by using the F-measure metric to pick the top- $K$  queries for issuing to the database— all of which have the same number of attributes constrained. A more effective way to trade off precision with the throughput of the rewritten queries is by making an "online" decision on the number of attributes to be constrained. We propose a technique, *BN-Beam* which searches over the markov blanket of the original query-constrained attribute, and picks the best subset of the attributes with high precision and throughput.

*3.1.1 Generating Rewritten Queries using BN-Beam*

We now describe BN-Beam, our technique for generating rewritten queries which finds a subset of the attributes in the markov blanket of the query-constrained attribute with high precision and throughput. To illustrate how rewritten queries are generated using BN-Beam, consider the same query  $(Q) \sigma_{Body=Sedan}$ . First,  $Q$  is sent to the database to retrieve the base result set. We consider the attributes in the markov blanket of the query-constrained attribute to be the potential attributes on which the new queries will be formulated. We call this set the *candidate attribute set*.

For query  $Q$ , the candidate attribute set consists of attributes in the markov blanket of attribute  $Body$  which consists of attributes  $\{Model, Year\}$  for the Bayes Network in Figure 2.1.1. Once the candidate attribute set is determined, a beam search with a beam width,  $K$  and depth,  $L$ , is performed over the distinct value combinations in the base result set of the attributes in the candidate attribute set. For example, when the query  $Q$  is sent to the database fragment shown in Table 3.1, tuples  $t_1, t_3, t_4$  and  $t_5$  are retrieved. The values over which the search is performed for  $Model$  is  $\{A8, tl, 745\}$  and for  $Year$  is  $\{2002, 2003, 2005\}$ . Starting from an empty rewritten query, the beam search is performed over multiple levels, looking to expand the partial query at the previous level by adding an attribute-value to it. For example, at the first level of the search five partial rewritten queries:  $\sigma_{Model=745}$ ,  $\sigma_{Model=A8}$ ,  $\sigma_{Model=tl}$ ,  $\sigma_{Year=2002}$  and  $\sigma_{Year=2003}$  may be generated. An important issue here is to decide which of the queries should be carried over to the next level of search. Since there is a limit on the number of queries that can be issued to the autonomous database and we want to generate rewritten queries with high precision and throughput while keeping query processing costs low, we pick the top- $K$  queries based on the F-measure metric, as described earlier. The advantage of performing a search over both attributes and values for generating rewritten queries is that there is much more control over the throughput of the rewritten queries as we can decide how many attributes will be constrained.

The top- $K$  queries at each level are carried over to the next level for further expansion. For example, consider query  $\sigma_{Model=745}$  which was generated at level one. At level two, we try to create a conjunctive query of size two by constraining the other attributes in the candidate attribute set. Say we try to add attribute  $Year$ , we search over the distinct values of  $Year$  in the base set with attribute  $model$  taking the value  $745$ . At each level  $i$ , we will have the top- $K$  queries with highest F-measure values with  $i$  or fewer attributes constrained. The top- $K$  queries generated at the Level  $L$  are sorted based on expected precision and sent to the autonomous database in that order to retrieve the relevant possible answers.

We now describe the BN-Beam algorithm for generating rewritten queries for single-attribute queries.

In step 2(d), it is important to remove duplicates from  $\widehat{RS}(Q)$ . Since rewritten queries may constrain different attributes, the same tuple might be retrieved by different rewritten queries. For example, consider two rewritten queries-  $Q'_1: \sigma_{Model=A8}$  and  $Q'_2: \sigma_{Year=2005}$ , that can be generated at level one for the same user query  $Q$ , that aims to retrieve all *Sedan* cars. All *A8* cars manufactured in *2005* will be returned in the answer sets of both queries. Therefore,

Let  $R(A_1, A_2, \dots, A_n)$  be a database relation. Suppose  $MB(A_p)$  is the set of attributes in the markov blanket of attribute  $A_p$ . All the steps in processing a query  $Q: \sigma_{A_p=v_p}$  is the same as described for BN-All-MB except step 2(a) and 2(d).

2(a) Generate Rewritten Queries. A beam search is performed over the attributes in  $MB(A_p)$  and the value for each attribute is limited to the distinct values for each attribute in  $RS(Q)$ . Starting from an empty rewritten query, a partial rewritten query ( $PRQ$ ) is expanded, at each level, to add an attribute-value pair from the set of attributes present in  $MB(A_p)$  but not added to the partial rewritten query already. The queries with top- $K$  values for F-measure scores, computed from the estimated precision and estimated recall computed from the sample, are carried over to the next level of the search. The search is repeated over  $L$  levels.

2(d) Post-filtering. Remove the duplicates in  $\widehat{RS}(Q)$ .

### Algorithm 2: Algorithm for BN-Beam

we need to remove all duplicate tuples.

#### 3.1.2 Handling Multi-attribute Queries

**BN-All-MB:** The method described to handle single-attribute queries using BN-All-MB can be easily generalized to handle multi-attribute queries. The rewritten queries generated will constrain every attribute in the union of the markov blanket of the constrained attributes.

**BN-Beam:** Similarly, using BN-Beam to handle multi-attribute queries is simple extension of the method described for single-attribute queries. Recall that when the original query constrains a single attribute, the candidate attribute set, over which the search is performed consists of the attributes in the markov blanket of the constrained attribute. When there are multiple constrained attributes in the original query, the union of the attributes in the markov blanket of all the constrained attributes form the candidate attribute set. It is important to note that some of the attributes constrained in the original query may be contained in the markov blanket of the other constrained attributes. To retrieve relevant tuples with missing values on query-constrained attributes, those attributes should not be constrained in any of the rewritten queries. Therefore, when a query-constrained attribute is part of the markov blanket of another query-constrained attribute, it should be removed from the candidate attribute set. Retrieving relevant uncertain answers for multi-attribute queries has been only superficially addressed in the QPIAD system. It attempts to retrieve only uncertain answers with missing values on any one of the multiple query-constrained attributes. It does not retrieve tuples with missing values on multiple query-constrained attributes.

To illustrate how new queries are reformulated to retrieve possibly relevant answers with multiple missing values on query-constrained attributes, consider an example query

$\sigma_{Make = BMW \wedge Mileage = 40000}$  sent to database fragment in Table 3.1. First, this query retrieves the base result set which consists of tuples  $t_4, t_9, t_{10}$ . The set of candidate attributes on which the new queries will be formulated is obtained by the union of attributes in the markov blanket of the query-constrained attributes. For the learned Bayesian network shown in Figure 2.1.1, this set consists of  $\{Model, Year\}$ . Once the candidate attribute set is determined, a beam search with a beam width,  $K$ , is performed similar to the case when a single attribute is constrained. At the first level of the search some of the partial rewritten queries that can be generated are  $\sigma_{Model=745}$ ,  $\sigma_{Model=645}$  and  $\sigma_{Year=1999}$ . The top- $K$  queries with highest F-measure values are carried over to the next level of the search. The top- $K$  queries generated at the Level  $L$  are sent to the autonomous database to retrieve the relevant possible answers.

### 3.2 Empirical Evaluation of Query Rewriting

The aim of the experiments reported in this section is to compare the precision and recall of the relevant uncertain results returned by rewritten queries generated by AFDs and Bayes networks for single and multi-attribute queries. To evaluate the quality of rewritten queries generated for single-attribute queries, we compare BN-All-MB and AFDs. As we will discuss in Section 3.2.1 we find that the performance of BN-All-MB and AFDs is about the same. We then compare BN-All-MB and BN-Beam when there is a limit on the number of queries that can be issued to the autonomous database. We discuss in Section 3.2.2 that BN-Beam does a better job of trading precision with throughput compared to BN-All-MB. Finally, we compare BN-Beam and AFD-based approaches for multi-attribute queries when there is a limit on the number of queries that can be issued.

As mentioned before, we use a car database extracted from *Cars.com* [2] with a schema  $Cars(Model, Year, Body, Make, Price, Mileage)$  consisting of 55,000 tuples. The second database used is  $Adult(WorkClass, Occupation, Education, Sex, HoursPerWeek, Race, Relationship, NativeCountry, MaritalStatus, Age)$  consisting of 15,000 tuples obtained from UCI [12] data repository. These datasets are partitioned into test and training sets. In most information integration scenarios having access to sufficient training data is often very expensive as it involves sampling the autonomous data sources, which is costly. Therefore, only 15% of the tuples are used as the training set. The training set is used for learning the topology and parameters of the Bayes network and AFDs. It is also used for estimating the expected selectivity of the rewritten queries. We use the Expectation Propagation inference algorithm [11] (with 10 samples) available in INFER.NET software package [10] for carrying inference on the

Bayes network.

In order to evaluate the relevance of the answers returned, we create a copy of the test dataset which serves as the *ground truth* dataset. We further partition the test data into two halves. One half is used for returning the certain answers, and in the other half all the values for the constrained attribute(s) are set to null. Note that this is an aggressive setup for evaluating our system. This is because typical databases may have less than 50% incompleteness and even the incompleteness may not be on the query-constrained attribute(s). The tuples retrieved by the rewritten queries from the test dataset are compared with the ground truth dataset to compute precision and recall. Since the answers returned by the certain result set will be the same for all techniques, we consider only uncertain answers while computing precision and recall.

### 3.2.1 Comparison of Rewritten Queries Generated by AFDs and BN-All-MB

Figure 3.1 shows the precision-recall curve for queries on attribute *Make* in the Car database. The size of the markov blanket and the determining set is one for attribute *Make*. We note that there is no difference in the quality of the results returned by AFDs and BN-All-MB in this case (See Figure 3.1). Next, we compare the quality of the results returned by AFDs and BN-All-MB when the size of the markov blanket and determining set of the AFD of the constrained attribute is greater than one. Figure 3.2 shows the precision and recall curves for a queries issued to car database, and Figure 3.3 shows the curves for a query on Adult database. For the query on the Adult database, we found that the order in which the rewritten queries were ranked were exactly the same. Therefore, we find that the precision-recall curves of both the approaches lie one on top of the other. For the queries issued to the car database, we find that there are differences in the order in which the rewritten queries are issued to the database. However, we note that there is no clear winner. The curves lie very close to each other, alternating as the number of results returned increases. Therefore the performance of AFDs and BN-All-MB is comparable for single-attribute queries.

### 3.2.2 Comparison of Rewritten Queries Generated by BN-All-MB and BN-Beam

As we mentioned earlier, the issue with constraining every attribute in the markov blanket is that the size of these sets could get arbitrarily large. When that happens, the number of queries that need to be sent to the database increases significantly. To alleviate this problem, we proposed a technique, BN-Beam, which performs a search over both attributes and their values. Figure 3.4 shows the increase in recall of the results for three different values of  $\alpha$  in the F-measure metric



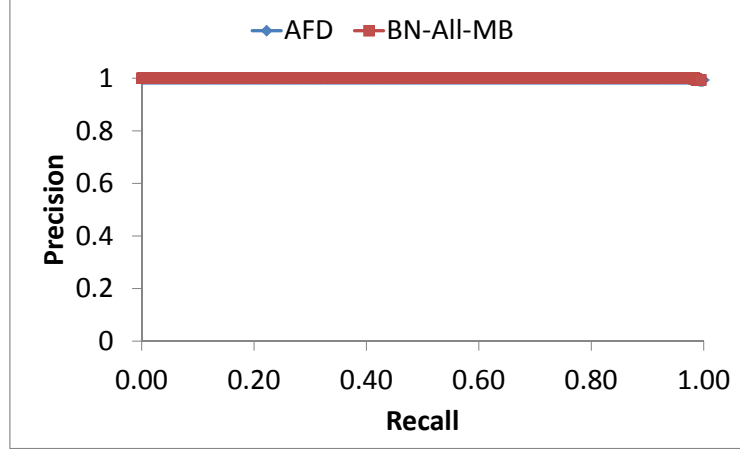


Figure 3.1: Precision-Recall curve for query  $\sigma_{Make}$

when ten queries can be issued to the database. We refer to results for different values for  $\alpha$  for BN-Beam as BN-Beam- $\alpha$  (substitute  $\alpha$  with its value) and BN-All-MB- $\alpha$  for BN-All-MB. For BN-Beam, the level of search,  $L$ , is set to two. We note that there is no change in recall for all the three cases for BN-All-MB. This is because there are no rewritten queries with high throughput, therefore just increasing the  $\alpha$  value does not increase recall. For BN-Beam, we see that the recall increases with increase in the value of  $\alpha$ . Figure 3.5 shows the change in precision for different values of  $\alpha$  as the number of queries sent to the database increases. As expected, the precision of BN-Beam-0.30 is higher than BN-Beam-0.35 and BN-Beam-0.40. In particular, we point out that the precision of BN-Beam-0.30 remains competitive with BN-All-MB-0.30 in all the cases while providing significantly higher precision. BN-Beam is able to retrieve relevant incomplete answers with high recall without any catastrophic decrease in precision.

### 3.2.3 Comparison of Multi-attribute Queries

In this section we compare Bayes network and AFD approaches for retrieving relevant uncertain answers with multiple missing values when multi-attribute queries are issued by the user. We note that the current QPIAD system retrieves only uncertain answers with at most one missing value on query-constrained attributes. We compare BN-Beam with two baseline AFD approaches.

**1. AFD-All-Attributes:** The first approach that we call *AFD-All-Attributes* creates a conjunctive query by combining the best rewritten queries for each of the constrained attributes. The best rewritten queries for each attribute constrained in the original query are computed inde-

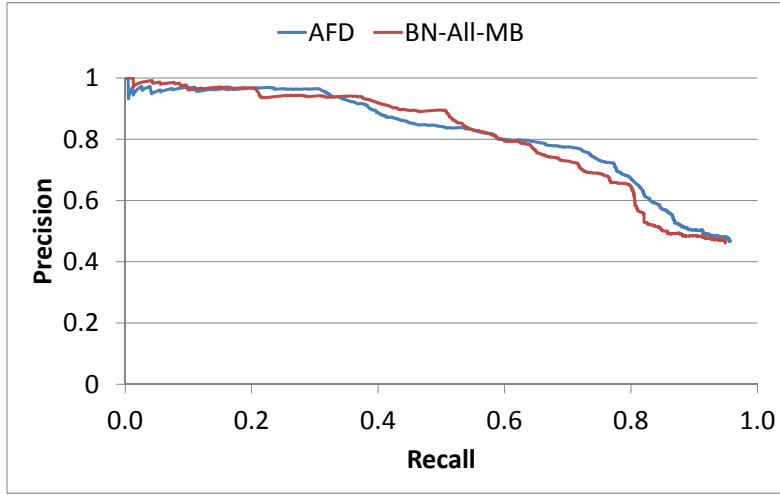


Figure 3.2: Precision-recall curve for query  $\sigma_{Body}$  averaged over 3 queries

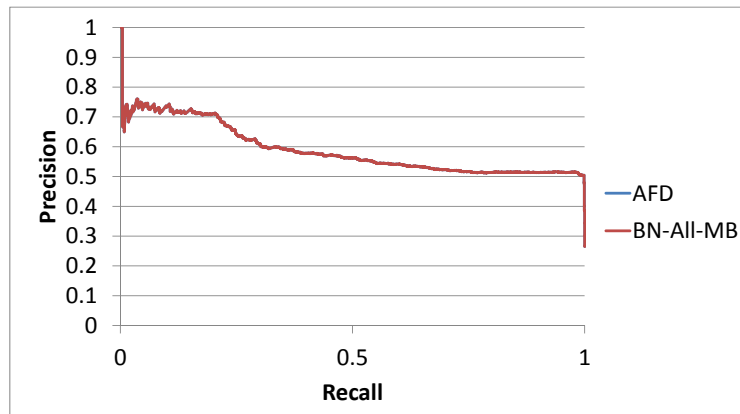


Figure 3.3: Precision-Recall curve for query  $\sigma_{Relationship=Not-in-family}$

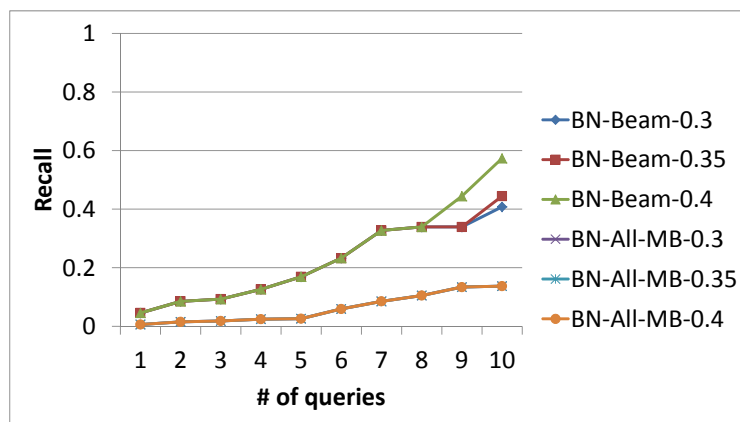


Figure 3.4: Change in recall for different values of  $\alpha$  in F-measure metric for top-10 rewritten queries for  $\sigma_{Year} = 2002$

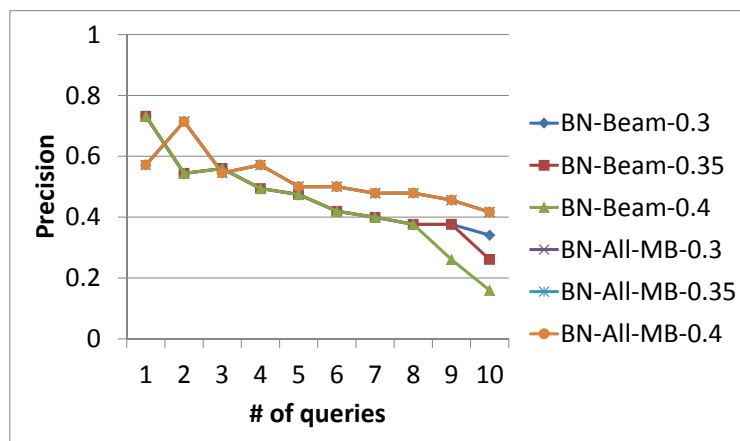


Figure 3.5: Change in precision for different values of  $\alpha$  in F-measure metric for top-10 rewritten queries for  $\sigma_{Year} = 2002$

pendently and new rewritten queries are generated by combining a rewritten query for each of the constrained attributes. The new queries are sent to the autonomous database in the decreasing order of the product of the expected precision of the individual rewritten queries that were combined to form the query. AFD-All-Attributes technique is only used for multi-attribute queries where the determining set of each of the attributes are disjoint.

**2. AFD-Highest-Confidence:** The second approach the we call, *AFD-Highest-Confidence* uses only the AFD of the query-constrained attribute with the highest confidence for generating rewritten queries, ignoring the other attributes.

We evaluate these methods for selection queries with two constrained attributes. For BN-Beam, the level of search is set to 2 and the value for  $\alpha$  in the F-measure metric is set to zero.

### 3.2.3.1 Comparison of AFD-All-Attributes and BN-Beam

Figure 3.6 shows the precision-recall curve for the results returned by top ten rewritten queries by AFD-All-Attributes and BN-Beam for the query  $\sigma_{Make=bmw \wedge Mileage=15000}$  issued to the Cars database. Figure 3.8 shows a similar curve for the query  $\sigma_{Education=HS-grad \wedge Relationship=Husband}$  issued to the Adult database. We note that the recall of the results returned by AFD-All-Attributes is significantly lower than BN-Beam in both cases (see figures 3.7 and 3.9). This is because the new queries generated by conjoining the rewritten queries of each constrained attribute do not capture the joint distribution of the multi-attribute query. Therefore, the throughput of these queries are often very low, in the extreme case they even generate empty queries. The precision of the results returned by AFD-All-Attributes is only slightly higher than BN-Beam (See Figures 3.6 and 3.8). By retrieving answers with a little lesser precision and much higher recall than AFD-All-Attributes, BN-Beam technique becomes very effective in scenarios where the autonomous database has limits on the number of queries that it will respond to.

### 3.2.3.2 Comparison of AFD-Highest-Confidence and BN-Beam

Figure 3.10 and 3.12 show the precision-recall curves for the results returned by the top ten queries for multi-attribute queries issued to the Cars and Adult databases respectively. Figures 3.11 and 3.12 show the change in recall with each of the top 10 rewritten query issued to the autonomous database. We note that the recall of the results returned by AFD-Highest-Confidence is much higher than BN-Beam. However, this increase in recall is accompanied by a drastic fall in precision. This is because AFD-Highest-Confidence approach is oblivious to

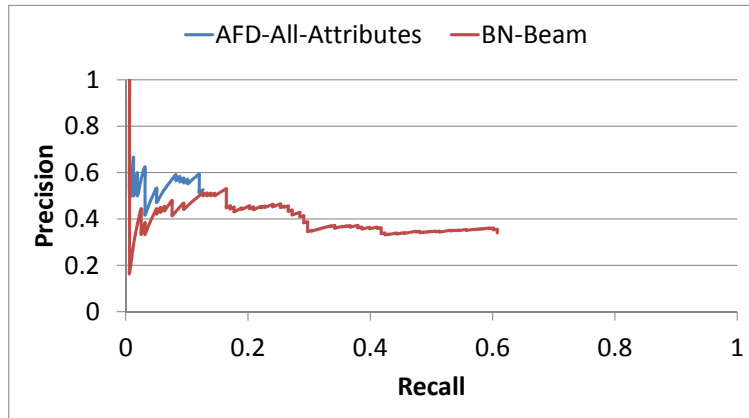


Figure 3.6: Precision-recall curve for the results returned by top-10 rewritten queries for the query  $\sigma_{Make=bmw \wedge Mileage=15000}$

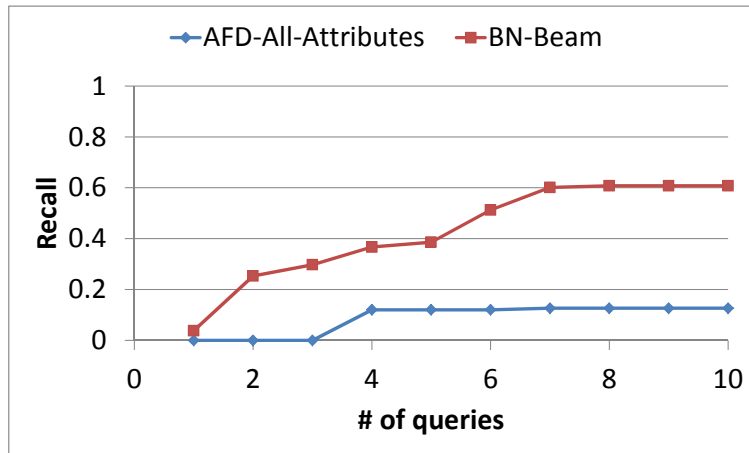


Figure 3.7: Change in recall as the number of queries sent to the autonomous database increases for the query  $\sigma_{Make=bmw \wedge Mileage=15000}$

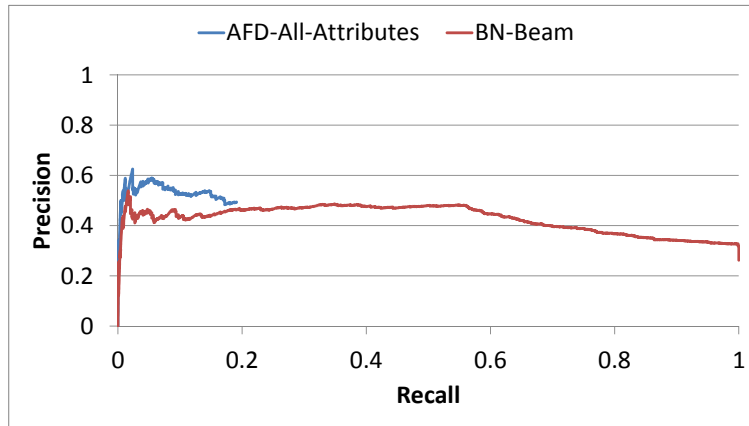


Figure 3.8: Precision-recall curve for the results returned by top-10 rewritten queries for the query  $\sigma_{Education=HS-grad \wedge Relationship=Husband}$

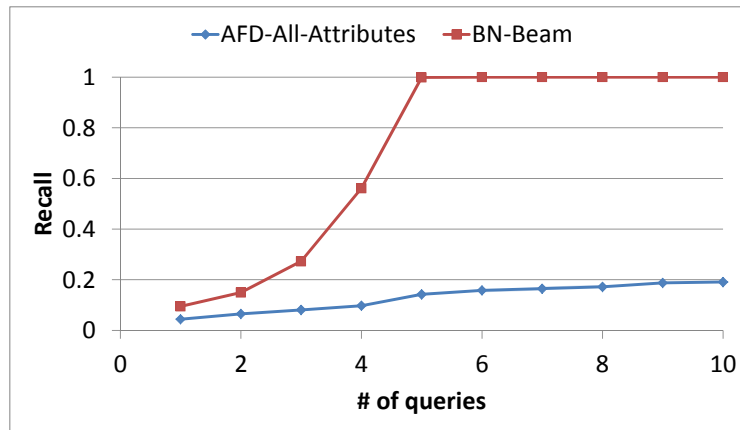


Figure 3.9: Change in recall as the number of queries sent to the autonomous database increases for the query  $\sigma_{Education=HS-grad \wedge Relationship=Husband}$

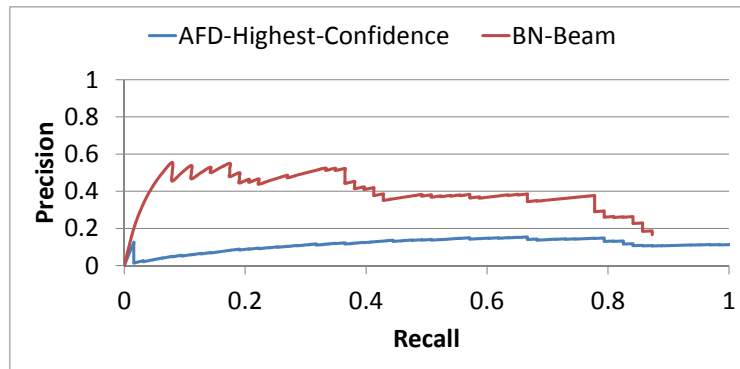


Figure 3.10: Precision-recall curve for the results returned by top-10 rewritten queries for the query  $\sigma_{Make=kia \wedge Year=2004}$

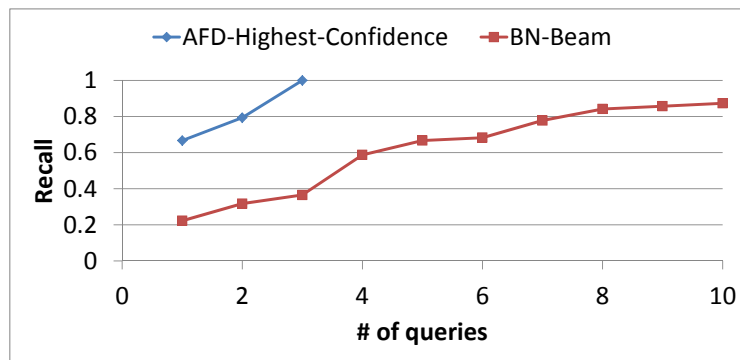


Figure 3.11: Change in recall as the number of queries sent to the autonomous database increases for the query  $\sigma_{Make=kia \wedge Year=2004}$

the values of the other constrained attributes. Thus, this approach too, is not very effective for retrieving relevant possible answers with multiple missing values for multi-attribute queries.

### 3.3 Summary

In this chapter, we described BN-All-MB, a technique for generating rewritten queries using a learned Bayes network model. We showed that for single-attribute queries, BN-All-MB and AFDs retrieve relevant uncertain answers with about the same precision and recall. Next, we motivated the need for novel techniques and proposed BN-Beam for reformulating queries

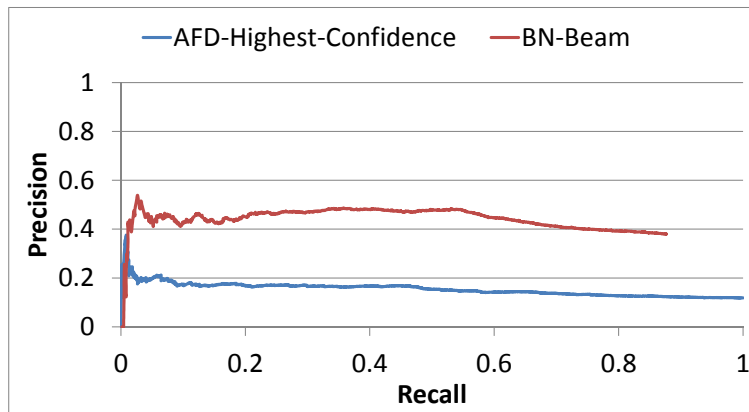


Figure 3.12: Precision-recall curve for the results returned by top-10 rewritten queries for the query  $\sigma_{WorkClass=private \wedge Relationship=own-child}$

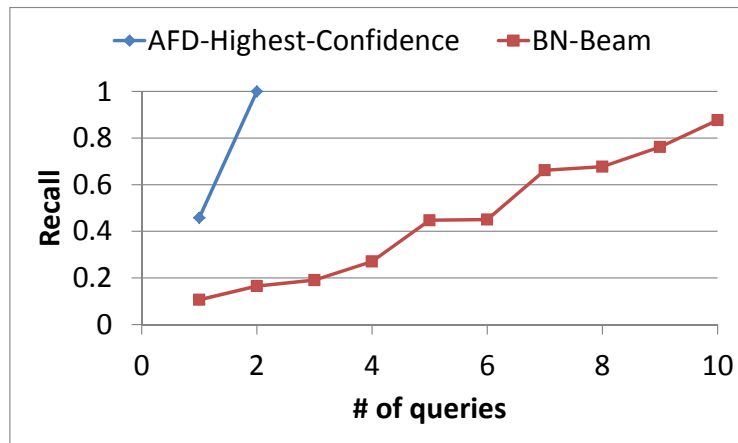


Figure 3.13: Change in recall as the number of queries sent to the autonomous database increases for the query  $\sigma_{WorkClass=private \wedge Relationship=own-child}$



that can carefully trade off precision with throughput when there are limits on the number of queries that can be issued to the autonomous database. We compared BN-Beam and BN-All-MB and showed that BN-Beam is adept at increasing recall without sacrificing too much on precision. We also described how BN-Beam can be extended to retrieve relevant uncertain answers with multiple missing values. Finally, we showed that BN-Beam trumps AFD-based approaches for retrieving uncertain tuples with multiple missing values when the autonomous database imposes limits on the number of queries it will answer.

### CONCLUSION & FUTURE WORK

#### 4.1 Conclusion

In this thesis, we presented a comparison of cost and accuracy trade-offs of using Bayes network models and Approximate Functional Dependencies (AFDs) for handling incompleteness in autonomous databases. We showed how a generative model of an autonomous database can be learnt and used by query processors while keeping costs manageable.

We compared Bayesian networks and AFDs for imputing single and multiple missing values. We showed that Bayes networks have a significant edge over AFDs in dealing with missing values on multiple correlated attributes and at high levels of incompleteness in test data.

Further, we presented a technique, BN-All-MB, for generating rewritten queries using Bayes networks. We then proposed a technique, BN-Beam, to generate rewritten queries that retrieve relevant uncertain results with high precision and throughput, which becomes very important when there are limits on the number of queries that autonomous databases respond to. We showed that BN-Beam trumps AFD-based approaches for handling multi-attribute queries. BN-Beam contributes to the QPIAD system by retrieving relevant uncertain answers with multiple missing values on query-constrained attributes with high precision and recall.

#### 4.2 Future Work

##### 4.2.1 Generating Rewritten Queries in the Presence of Limited Query Patterns

When the access to autonomous databases is through forms, the query patterns that are supported are often limited. Not all attributes present in the database are *constrainable*. Therefore the rewritten queries can only constrain a subset of the attributes in the database. For example, an autonomous database which stores data about used cars might not allow queries to specify the *Body* of the car. Therefore, no rewritten query issued to the autonomous database can constrain the attribute *Body*. The BN-Beam technique can be easily extended to generate rewritten queries under these scenarios. Instead of initializing the candidate attribute set to the attributes in the markov blanket (or union of the markov blanket for multi-attribute queries), we can initialize it to the set of all constrainable attributes. If this set has many attributes, we can consider only the constrainable attributes that are part of the markov blanket of the constrained attribute(s).

ID	Make	Model	Year	Body
1	Audi	A8	2005	Sedan
2	Audi	A8	2005	null
3	BMW	745	2002	Sedan
4	BMW	745	1999	null
5	null	A8	2005	null

Table 4.1: A Fragment of a car database

#### 4.2.2 Reducing CPU Time in Generating Rewritten Queries using BN-Beam

As the size of the candidate attribute set and their domains over which we search increases, the CPU time taken to generate the rewritten queries can increase considerably. One way to reduce the time spent in generating the rewritten queries is to choose the correct beam width( $K$ ) and the level( $L$ ) to which we search. Increasing the number of levels of the search increases the precision of the results returned and reduces the throughput of each query. On the other hand, increasing the width of the beam increases the number of queries that are generated, which increases the number of results returned. At level  $L$ , all the queries in the beam will have at most  $L$  attributes constrained. Setting the beam width to 1 and increasing the number of levels will generate a single rewritten query with high precision. On the other hand, setting the level of the search to 1 and increasing the beam width will produce single-attribute rewritten queries, which will have high throughput.

#### 4.2.3 Redundancy Metrics to Improve Recall

In BN-Beam, while picking the top- $K$  queries that are carried over to the next level of search, picking queries only based on relevance to the original user query does not consider whether a tuple has been retrieved already by previously issued rewritten queries. Using a redundancy metric in addition to the relevance metric will further improve the recall of the results returned by BN-Beam. To illustrate this point, consider a query  $Q : \sigma_{Body=Sedan}$  issued to the database fragment shown in Table 4.1. Tuples  $t_1, t_3$  are returned as the set of certain answers. Issuing a rewritten query  $Q'_1 : \sigma_{Model=A8}$  will retrieve uncertain tuples  $t_2, t_5$ . Now issuing a query  $\sigma_{Year=2005}$ , will only retrieve relevant uncertain tuples previously retrieved by  $Q'_1$ . Instead, issuing a rewritten query,  $\sigma_{Make=BMW}$ , will retrieve a tuple  $t_4$ , which was not returned previously. Therefore, using redundancy metrics will further improve the recall of the results returned by BN-Beam.

## REFERENCES

- [1] BANJO. <http://www.cs.duke.edu/amink/software/banjo/>.
- [2] Cars.com. <http://www.cars.com>.
- [3] CBioC. <http://www.cbio.org>.
- [4] Bayes Net Toolbox for Matlab. <http://code.google.com/p/bnt/>.
- [5] Rahul Gupta and Sunita Sarawagi. Creating probabilistic databases from information extraction models. In *VLDB*, pages 965–976, 2006.
- [6] David Heckerman. The certainty-factor model. *Encyclopedia of Artificial Intelligence*, Second Edition, 1992.
- [7] Hemal Khatri. QUERY PROCESSING OVER INCOMPLETE AUTONOMOUS WEB DATABASES. Master's thesis, Arizona State University, Tempe, USA, 2006.
- [8] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.
- [9] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.
- [10] T. Minka, J.M. Winn, J.P. Guiver, and D.A. Knowles. Infer.NET 2.4, 2010. Microsoft Research Cambridge. <http://research.microsoft.com/infernet>.
- [11] Thomas P. Minka. Expectation propagation for approximate bayesian inference. In *UAI*, pages 362–369, 2001.
- [12] UCI Machine Learning Repository. <http://archive.ics.uci.edu/ml/datasets.html>.
- [13] Stuart J. Russell and Peter Norvig. *Artificial Intelligence - A Modern Approach (3. internat. ed.)*. Pearson Education, 2010.
- [14] Ross D. Shachter. Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). In *UAI*, pages 480–487, 1998.
- [15] Garrett Wolf, Aravind Kalavagattu, Hemal Khatri, Raju Balakrishnan, Bhaumik Chokshi, Jianchun Fan, Yi Chen, and Subbarao Kambhampati. Query processing over incomplete autonomous databases: query rewriting using learned data dependencies. *VLDB J.*, 18(5):1167–1190, 2009.