7-16-2002

User-guided knowledge discovery using Bayesian networks.

Jie Deng

USER-GUIDED KNOWLEDGE DISCOVERY USING BAYESIAN NETWORKS

A Thesis

Presented to the

Department of Computer Science

and the

Faculty of the Graduate College

University of Nebraska

In Partial Fulfillment

of the Requirements for the

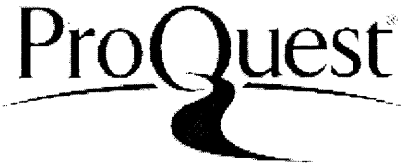
Degree of Master of Science

University of Nebraska at Omaha

By

Jie Deng

July 16, 2002



ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346

THESIS ACCEPTANCE

Acceptance for the faculty of the Graduate College, University of Nebraska, in partial fulfillment of the requirements for the degree of Master of Science, University of Nebraska at Omaha.

Committee

Name	Department/School
John Kennedy	Mathematics
Hesham Ali	Computer Science
Qiang Ye	Computer Science

Chairperson _____

Date 7/9/2002

USER-GUIDED KNOWLEDGE DISCOVERY USING BAYESIAN NETWORKS

Jie Deng (M.S.)

University of Nebraska, 2002

Advisor: Dr. Qiuming Zhu

A Bayesian network is a graphical model that encodes probabilistic relationships among variables of interest. When used in conjunction with statistical techniques, the graphical model has shown to be remarkably effective for some data-modeling problems. In this paper, we represent a computational model to apply Bayesian networks to knowledge discovery under uncertainty in a decision support system. Major features of this model include user-computer interaction and iterative information extraction. The user plays a primary role when determining the acceptance or refusal of intermediate information, while the computer in a supporting role crunches the numbers.

Two computation streams are provided in the model: (1) Top-down stream: the user enters the expectation value for the goal, and then calculates the expected values for all the nodes in the network. (2) Bottom-up stream: the user input provides evidence into the network, and testifies the effect of the evidence to the goal node.

We also designed and developed a software prototype to demonstrate the application of the proposed model. By using the software prototype, the user can easily construct and modify a Bayesian network. Not only does the network establish a

connection between the customer requirement and the given source data, but also serves as the tool for our knowledge discovery process. With a tentative Bayesian network, propagations are carried out to testify the relevance represented in the network. After reviewing the results, the user may decide to remove some irrelevant components from the network, or he may want to add new components into the network, which will start a new iteration of the knowledge discovery process. As the repetition goes on, the user will get closer and closer to reach a Bayesian network that suits the problem domain. The information retrieved by applying the derived Bayesian network, together with the network itself, will then be used in further decision support.

ACKNOWLEDGEMENTS

First, I would like to thank my advisor, Dr. Qiuming Zhu. His deep understanding of this project always led me to the right direction of the study. He also helped me set up an overall view of this study, and indicated the right way to organize my results into this thesis. I am glad to work under a smart and kind advisor who always suggests innovative and challenging ideas, yet also gives me the chance to try my own thoughts. I appreciate his kind help to my living, too. Thank you very much for the enjoyable dinners.

I would like to thank Dr. John Konvalina and Dr. Hassan Farhat, for giving me precious suggestions on this thesis and joining my committee. Thank you for your time and instructions, not only for this thesis, but also for the interesting classes that gave me so much knowledge and fun.

I would like to give my special thanks to Mr. Miao Chen, who gave me precious advices about implementing a software prototype. I would also like to extend my thanks to his wife, Mrs. Hongmei Cui. We always share our joy and sadness together. Your support as a close friend is something I can always count on.

I would like to thank all those who helped me toward the accomplishment of this thesis. Please accept my sincere appreciation.

Finally, and most important, I would like to thank my husband, Yongheng Huang, for your love and kindness. I am the luckiest woman in the world to have you as my husband. I love you.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 The Problem Statement

A Bayesian network is a graphical model for probabilistic relationships among a set of variables. It has attracted extensive attention in the last twenty years, especially recently, as a possible solution for reasoning uncertainty in artificial intelligence. Heckerman [28] applied Bayesian networks to Decision Trees. Indrawan etc. [33] used Bayesian networks as a text retrieval engine. More researches on Bayesian networks are underway and more techniques are evolving. Our research explores a new approach of applying Bayesian networks to knowledge discovery.

The unique feature of Bayesian networks in probability inference renders it a favorable tool in the implementation of knowledge discovery under uncertainty. Many successful cases of Bayesian networks applied in knowledge discovery in specific area have been reported recently. Ahn and Ezawa of AT&T Lab [1] demonstrated that by linking the BN learning model with rigorous decision-making techniques, such as influence diagrams, the decision support for real-time telemarketing operations was shown to provide an intelligent decision advice. Bayesian networks have also been proven useful in practical applications such as medical diagnosis and diagnosis of mechanical failures [30].

Despite of recent successes, the Bayesian networks applications in knowledge discovery are far from reaching their potential. Some especially challenging obstacles include: the incorporation of background and associated knowledge, human-computer

interaction, efficient inference algorithms, and so on. To further explore the potential of implementing Bayesian networks in knowledge discovery, we introduced a user-guided computational model in our research. This model uses Bayesian network propagation as the technique to discovery patterns from given data. The biggest advantages of this model include: (a) it enables easy user-computer communication, and (b) it provides an iterative discovering process. We also implemented a software prototype to demonstrate the procedure represented in the model.

1.2 User-Guided Versus Automated Knowledge Discovery Process

In general, two models can be used for knowledge discovery and data mining [47]:

- Automated knowledge discovery and data mining: The data mining system is responsible for automatically discovering knowledge from source data, by detecting patterns and correlations in the data.
- User-guided knowledge discovery and data mining: The user is involved directly in the process of knowledge discovery.

The automated knowledge discovery model seems to be powerful and attractive. However, so far, knowledge discovery and data mining largely remain an art, usually requiring much skill from the users. This picture is further complicated by the intrinsic problems of KDD (knowledge discovery in databases) and data mining, such as overabundance of knowledge patterns generated in the intermediate and final stages of the discovery, and poor integration of the KDD systems with the databases and decision

support systems. The automated knowledge discovery model, therefore, is more prone to suffer problems such as overabundance, because the user has little control (or no control at all) on the knowledge discovery process.

In contrast, in user-guided model for KDD and data mining, the user has primary responsibility for discovering rules, and source data plays a supporting role. Typically, the user makes up a hypothesis, and runs tests of the database to verify it against the database. Thus, there may be several iterations, involving successive refinement of the hypothesis, as the user tests more refined versions. This approach is suitable for data mining in organizations where the user control over the bulk of information is a primary concern. However, this approach is also more challenging and requires some kind of user training.

Our research explored a user-guided model. In this model, the user makes up a hypothesis according to the customer requirement. Tests are then carried out to verify or refute that hypothesis. Based on the information retrieved, the user may then come up with a new hypothesis, or may refine the existing hypothesis, and may verify it against the source data. This process will iterate many times. Eventually, meaningless information will be filtered out, and information that is more useful will get the focus.

1.3 Bayesian Networks as a New Knowledge Discovery Technique

There are numerous representations available for data mining, including rule bases, decision trees, and artificial neural networks; and there are many techniques for data mining, such as density estimation, classification, regression, and clustering. So,

what do Bayesian networks and Bayesian methods have to offer? There are at least three answers:

One, Bayesian networks allow one to learn about causal relationships. Learning about causal relationships is important for at least two reasons. The process is useful when we are trying to gain understanding about a problem domain, for example, during exploratory data analysis. In addition, knowledge of causal relationships allows us to make predictions in the presence of interventions. For example, a marketing analyst may want to know whether or not it is worthwhile to increase exposure of a particular advertisement in order to increase the sales of a product. To answer this question, the analyst can determine whether or not the advertisement is a cause for increased sales, and to what degree. The use of Bayesian networks helps to answer such questions even when no experiment about the effects of increased exposure is available.

Two, Bayesian networks in conjunction with Bayesian statistical techniques facilitate the combination of domain knowledge and data. Anyone who has performed a real-world modeling task knows the importance of prior or domain knowledge, especially when data is scarce or expensive. The fact that some commercial systems (i.e., expert systems) can be built from prior knowledge alone is a testament to the power of prior knowledge. Bayesian networks have a causal semantics that makes the encoding of causal prior knowledge particularly straightforward. In addition, Bayesian networks encode the strength of causal relationships with probabilities. Consequently, prior knowledge and data can be combined with well-studied techniques from Bayesian statistics.

Three, Bayesian methods in conjunction with Bayesian networks and other types of models offer an efficient and principled approach for avoiding the over fitting of data. The network, by itself, is a processed and simplified representation of the source data: The network structure reflects the relationship among database attributes or problem domain events, while the statistical distribution contains quantitative information obtained from source data.

1.4 Organization of the Thesis

The thesis is organized as follows:

Chapter 1 contains an introduction of the research problem, the comparison between user-guided KDD process and automated KDD process, and the reason for choosing Bayesian networks as a knowledge discovery technique. In Chapter 2, we will first briefly introduce the basic concepts of Bayesian networks. Following that, we will discuss in detail the algorithm we used for probability inference. Chapter 3 describes the knowledge discovery process in general. Chapter 4 contains the goal-driven interactive knowledge discovery model we developed, and Chapter 5 is an explanation of the software prototype we developed to demonstrate that model. Finally, chapter 6 contains a brief conclusion to the whole research and some suggestions for future work.

CHAPTER 2: BAYESIAN NETWORKS

2.1 Overview of Bayesian Networks

A Bayesian Network is a knowledge representation scheme dealing with probabilistic knowledge. They have been referred to by many other names, including belief networks, directed Markov fields, and causal probabilistic networks. In this section, we briefly discuss the typical features of Bayesian networks.

A Bayesian network for a set of variables $X = \{X_1, \ldots, X_n\}$ consists of: (1) a network structure S that encodes a set of conditional independence assertions about variables in X, and (2) a set P of local probability distributions associated with each variable. Together, these components define the joint probability distribution for X. The network structure S is a directed acyclic graph (DAG). Each of its nodes is a domain variable that can take on a set of discrete values, or in some cases of a continuous value. For the purpose of this thesis, we will be confining our discussion to discrete random variables with finite sample spaces. The problems we treat in this paper are all amenable to representation in this form, and the algorithms we use for evaluating such networks are best suited to networks of such random variables. The arcs in structure S represent direct influences between random variables. It is conventional, but not necessary, to assign direction to the arcs in agreement with intuitions about causality.

To give a more visually direct explanation, let us take an example for consideration. Figure 2-1 shows a Bayesian network used to solve a *family-out* problem [7]. The *family-out* problem is described as the following:

Mr. Thomas wants to know if his family is home before he tries the doors. Often when his wife leaves the house, she turns on an outdoor light. However, she sometimes turns on this light if she is expecting a guest. Also, they have a dog. When nobody is home, the dog is put in the back yard. The same is true if the dog has bowel troubles. Finally, if the dog is in the backyard, Mr. Thomas will probably hear her barking, but sometimes he can be confused by other dogs barking.

There are five random variables: *family-out*, *light-on*, *bowel-problem*, *dog-out* and *hear-bark*. These variables are all Boolean — their states are either true or false. The diagram in Figure 2-1 represents the qualitative information that *family-out* influences *light-on* and *dog-out*, that *bowel-problem* influences *dog-out*, and that *dog-out* influences *hear-bark*. Furthermore, it also represents the assumptions that *hear-bark* is conditionally independent of *family-out*, given *dog-out*, and that *hear-bark* is conditionally independent of *bowel-problem*, given *dog-out*, etc.

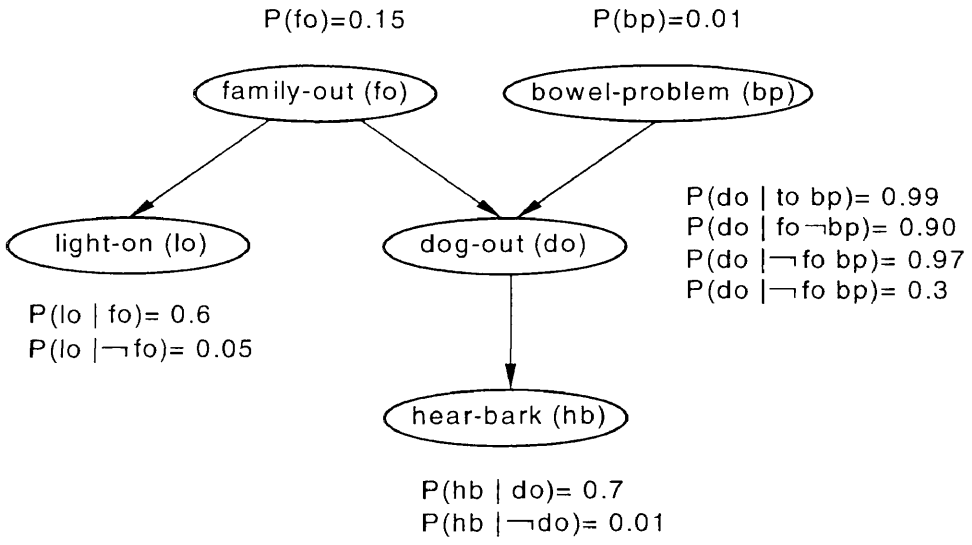


Figure 2-1. A Bayesian network modeling the *family-out* problem

Because Bayesian networks allow us to capture information about conditional independence, they make it possible to compactly represent probability distributions. The probability distribution corresponding to a Bayesian network may be specified by giving conditional probability distributions for each node in the network. The conditional probability distribution gives the probability of the random variable taking on each possible value conditioned on each combination of values of its parent nodes. If the node has no parents (a "root" node), one specifies a prior distribution over the values of that node. For Example, in order to specify a probability distribution among *family-out*, *bowel-problem* and *dog-out*, one would need the following:

P(family-out = true)	P(family-out = false)
P(bowel-problem = true)	P(bowel-problem = false)
P(dog-out = true | family-out = true, bowel-problem = true)	P(dog-out = false | family-out = true, bowel-problem = true)
P(dog-out = true | family-out = true, bowel-problem = false)	P(dog-out = false | family-out = true, bowel-problem = false)
P(dog-out = true | family-out = false, bowel-problem = true)	P(dog-out = false | family-out = false, bowel-problem = true)
P(dog-out = true | family-out = false, bowel-problem = false)	P(dog-out = false | family-out = false, bowel-problem = false)

Figure 2-2. The probability distribution among nodes *family-out*, *bowel-problem* and *dog-out* in *family-out* problem

The conditional probability distributions in the *family-out* problem are also given in Figure 2-1. For example, it states that if family members leave the house, they will turn on the outside light 60 percent of time, but the light will be turned on even when they do not leave 5 percent of time (say, because someone is expected). As you may notice in Figure 2-1, there are a big number of parameters (conditional probability

distributions) required by Bayesian networks. You may wonder how the numbers are obtained. In the demonstration given in Chapter 5, they are all made up just for the purpose of demonstration of the knowledge discovery process. But even in the real world application, there is really nothing wrong with making up numbers. For one thing, experts are fairly good at it. In one study [48], doctor's assessments of the numbers required for a Bayesian network were compared to the numbers that were subsequently collected and found to be pretty close. In a few cases, of course, it might actually be possible to collect data and produce the required numbers. When this is possible, we have the ideal case.

Bayesian networks allow one to calculate the conditional probabilities of the nodes in the network given that the values of some of the nodes have been observed. Considering the *family-out* example, if I observe that the light is on (*light-on* = *true*) but do not hear my dog (*dog-bark* = *false*), I can calculate the conditional probability of *family-out* given these pieces of evidence. In general, the computation of a probability of interest given a model is known as probabilistic inference. The following section discusses the probabilistic inference in Bayesian networks.

2.2 Inference Algorithms

2.2.1 General discussion on the inference algorithms

As noted above, the basic computation on Bayesian network is the computation of every node's belief (probability). Cooper has proved that this computation is NP-hard [12]. An efficient, exact algorithm does not exist to compute probabilistic inference in

Bayesian networks. Probably this is the most important constraint on the use of Bayesian networks in general. For many applications, however, the networks are small enough (or can be simplified sufficiently) so that these complexity results are not fatal.

Current algorithms for probabilistic inference on belief networks can be placed into two categories: the exact methods, which deterministically calculate posterior probabilities; and approximate solutions, which often improve run times but have less accuracy. The most well known approximate methods are based on simulation. Dagum and Cooper [12] presented one such simulation algorithm for doing probabilistic inference in Bayesian networks. There are a bewilderingly large variety of variants of this scheme. All these methods, however, suffer from three problems: (1) the basic approximation problem (for example, the problem of determining if a probability is less than a specified value) is NP-hard; (2) error decreases as the square of the number of samples; and (3) unexpected evidence on non-root nodes can reduce the number of useful samples collected.

Although the approximate algorithm can, in some circumstances, perform efficient approximate inference in large and richly interconnected models, we will only use exact methods for our probabilistic inference. The major concern for avoiding approximate algorithms comes from two aspects: (1) above shortcomings of the approximate solutions are not negligible in our perspective. Accuracy is required to assure that discovered knowledge is valid; (2) The run-time efficiency is not the major concern in our research since the networks we will build are simple enough to be handled quickly even with a non-efficient inference algorithm.

Among the exact solutions, a message-passing scheme proposed by Pearl [40] is the most widely accepted model for singly connected Bayesian Networks. This algorithm updates the probability distributions for each node in response to observations of one or more variables. It is guaranteed to converge to the correct posterior probabilities in graphic models where there can be at most one chain between any two variables. This algorithm is described in section 2.2.3, and has been adopted in our research as the propagation algorithm. Propagation in loopy graphs (by which we mean non-singly connected networks or graphs that more than one chain can exist to connect two vertices), however, still requires further study before we can agree on a unanimously efficient solution. Some approaches so far raised by researchers include conditioning method, clustering method and so on.

2.2.2 Introduction to Pearl's algorithm

When presenting the algorithm, we use the following denotation to present a Bayesian network:

$C = (V, E, P)$ V: the union of vertices in the graph

E: the union of edges in the graph

P: the probabilities in the associated probability table

Moreover, upper-case letters have been used to represent sets of events, and corresponding lower-case letters the single possible variables of the individual events.

For example, if we use A to represent a "*family-out*" event. Then for the two possible variables of this event, "*family is out*" or "*family is in*", letter "a_1" and "a_2" are used correspondingly. Conditional probability is denoted in a form like $P(m \mid n)$. $P(m \mid n)$ is the probability that event m could happen given the pre-assumption that event n has happened already. For instance, $P(b_1|a_1) = 0.7$ simply means if a_1 for sure happened, then the probability b_1 will happen 70 percent of time.

Below gives some other denotations we will use when describing the algorithm:

b_i : the ith possible value of node B

W: a subset of variables that are instantiated

W_B^-: the subset of W in the tree root at B

W_B^+ : $W - W_B^-$

Before we turn over to the steps of the algorithm, let us first introduce two important parameters: λ and π. Pearl's algorithm adopts a message-passing scheme to update the probability distributions. For each node in the network, there is a λ value and a π value associated. The node probability is proportional to both values. There are two other parameters associated with each link in the network: a λ message and a π message. The reason we call them messages is that their values are obtained from information about one of the adjacent nodes and are meaningful to the other node on the edge. For instance, a λ message is obtained from a child node and is used to propagate information

from the child node to the parent node. π messages work the other way. The definitions of λ and π are given below:

Definition for a node's λ and π values: Let $C = (V, E, P)$ be a Bayesian network, let W be a subset of instantiated variables, and let $B \in V$ have k possible values. Then we define for $1 \leq i \leq k$,

$$\lambda(b_i) = \begin{cases} P(W_B^- \mid b_i) & \text{if } B \notin W \\ 1 & \text{if } B \in W \text{ and } b_i \text{ is the instantiated value} \\ 0 & \text{if } B \in W \text{ and } b_i \text{ is not the instantiated value} \end{cases}$$

$$\pi(b_i) = P(b_i \mid W_B^+)$$

The entire vector of values, $\lambda(b_i)$ for $1 \leq i \leq k$, is called B's λ value and is denoted $\lambda(B)$. Likewise, the vector of values, $\pi(b_i)$ for $1 \leq i \leq k$, is called B's π value and is denoted $\pi(B)$.

Definition for λ messages on the edges: Let $C = (V, E, P)$ be a Bayesian network in which the graph is a tree, W a subset of instantiated variables, $B \in V$ have k possible values, and $C \in s(B)$ a child of B with m possible values. Then for $1 \leq i \leq k$, we define:

$$\lambda_C(b_i) = \sum_{j=1}^{m} P(c_j \mid b_i)\lambda(c_j)$$

The entire vector of values, $\lambda_C(b_i)$ for $1 \leq i \leq k$, is called the λ message from C to B and is denoted $\lambda_C(B)$.

Definition for π messages on the edges: Let $C = (V, E, P)$ be a Bayesian network in which the graph is a tree, W a subset of instantiated variables, $B \in V$ a variable which is not the root, and $A \in V$ the father of B. Suppose A has m possible values. Then we define for $1 \leq j \leq m$,

$$\pi_B(a_j) = \begin{cases} 1 & \text{if } A \text{ is instantiated for } a_j \\ 0 & \text{if } A \text{ is instantiated, but not for } a_j \\ \pi(a_j) \prod_{\substack{C \in s(A) \\ C \neq B}} \lambda_C(a_j) & \text{if } A \text{ is not instantiated} \end{cases}$$

The entire vector of values, $\pi_B(a_j)$ for $1 \leq j \leq m$, is called the π message from A to B and is denoted $\pi_B(A)$.

The term "*instantiate*" used in the above definition means given evidence. For example, if we instantiate node *light-on*'s first value b_1 in the *family-out* example, that means b_1 (which is "*light is on*") happened. Every time we instantiate an event, only one of its possible values will be instantiated due to mutual exclusiveness among them. The term "*instantiate*" used throughout the paper all means the same.

2.2.3 Operative formulas used in Pearl's algorithm

Below we list the operative formulas that will be used in the Pearl's algorithm. We will elaborate the algorithm in section 2.2.4 and then give the pseudocode we used to implement the algorithm in section 2.2.5.

Operative Formulas:

1. If B is a child of A, B has k possible values, A has m possible values, and B has one other parent D, with n possible values, then for $1 \le j \le m$ the λ message from B to A is given by

$$\lambda_B(a_j) = \sum_{p=1}^{n} \pi_B(d_p)\left(\sum_{i=1}^{k} P(b_i \mid a_j, d_p)\lambda(b_i)\right).$$

2. If B is a child of A and A has m possible values, then for $1 \le j \le m$ the π message from A to B is given by

$$\pi_B(b_i) = \begin{cases} 1 & \text{if } A \text{ is instantiated for } a_j \\ 0 & \text{if } A \text{ is instantiated , but not for } a_j \\ \dfrac{P'(a_j)}{\lambda_B(a_j)} & \text{if } A \text{ is not instantiated.} \end{cases}$$

where $P'(a_j)$ is defined to be the current conditional probability of a_j based on the variables thus far instantiated.

3. If B is a variable with k possible values, $s(B)$ is the set of B's Children, then for $1 \le j \le k$ the λ value of B is given by

$$\lambda(b_i) = \begin{cases} \prod_{C \in s(B)} \lambda_C(b_i) & \text{if } B \text{ is not instantiated} \\ 1 & \text{if } B \text{ is instantiated for } b_i \\ 0 & \text{if } B \text{ is instantiated, but not for } b_i. \end{cases}$$

4. If B is a variable with k possible values and exactly two parents, A and D, A has m possible values, and D has n possible values, then for $1 \le i \le k$ the π value of B is given by

$$\pi(b_i) = \sum_{j=1}^{m} \sum_{p=1}^{n} P(b_i \mid a_j, d_p) \pi_B(a_j) \pi_B(d_p).$$

5. If B is a variable with k possible values, then for $1 \le i \le k$, $P'(b_i)$, the conditional probability of b_i based on the variables thus far instantiated, is given by

$$P'(b_i) = \alpha \lambda(b_i) \pi(b_i).$$

2.2.4 Pearl's inference algorithm procedure

Pearl's inference algorithm consists of two parts: initialization and updating. In initialization, the Bayesian network is first initialized to compute the priori probabilities (i.e., the probabilities based on the instantiation of no variables) of all variables. When a variable is instantiated, or a λ or π message is received by a variable, one of the updating procedures will be triggered. The initialization and updating procedures are given as follows:

Initialization:

A. Set all λ values, λ messages, and π messages to 1.

B. For all roots A, if A has m possible values, then for $1 \leq j \leq m$, set

$$\pi(a_j) = P(a_j).$$

C. For all roots A for all children B of A, do

Post a new π message to B using operative formula 2.

{A propagation flow will then begin due to updating procedure C.}

Updating:

A. If a variable B is instantiated for b_j, then

Begin

1. Set $P'(b_j)=1$ and for $i \neq j$ set $P'(b_i)=0$;
2. Compute $\lambda(B)$ using operative formula 3;
3. Post new λ messages to all B's parent using operative formula 1;
4. Post new π messages to all B's children using operative formula 2

End.

B. If a variable B receives a new λ message from one of its children, and if B is not already instantiated, then

Begin

1. Compute the new value of $\lambda(B)$ using operative formula 3;
2. Compute the new value of $P'(B)$ using operative formula 5;
3. Post new λ messages to all B's parent using operative formula 1;

4. Post new π messages to B's other children using operative formula 2

End.

C. If a variable B receives a new π message from a parent, then

Begin

If B is not already instantiated, then

Begin

1. Compute the new value of $\pi(B)$ using operative formula 4;
2. Compute the new value of $P'(B)$ using operative formula 5;
3. Post new π messages to all B's children using operative formula 2

End;

If $\lambda(B) \neq (1, 1, \ldots, 1)$, then

4. Post new λ messages to B's other parents using operative formula 1

End.

2.2.5 Pseudocode for Pearl's inference algorithm

Regardless of the detailed formulas, following is a pseudocode we developed to realize this algorithm in C++ language.

```
Set initial λ values, λ messages, and π messages to 1
Input root node expectation
For all root nodes
{
        current node = root node
        while (current node is not a leaf node)
```

```
                Post a new π message to all the children of current node
                Update the probability of the children node
                Current node = children node
}

Instantiate events
For all the instantiate events
{
        Current node – instantiate node
        While(current node is not a instantiated node or not a root node)
                Post a new λ message to all its parents
                Compute the new probability value and λ value of the parent
                For all the other children of the parent node
                        Post a new π message
                        Do
                        {
                                update the probability and π value of the children node
                                post a new π message to the children of the children node
                        }Until (reaches an instantiate node or a leaf node)
                current node = parent node

        current node = instantiate node
        while(current node is not an instantiate node or a leaf node)
                post a new π message to all its children
                compute the new probability value and π value of the children
                for the other parents of the children
                        post new λ message
                        Do
                        {
                                update the probability and λ value of the children node
                                post a new λ message to the parent of the parent node
                        }Until (reaches an instantiate node or a root node)
                current node = children node
}
```

2.3 Construction of Bayesian Networks

Although researchers have made substantial advances in developing the theory and application of Bayesian Networks, the actual construction of these networks still remains a difficult and time-consuming task. The task is time-consuming because typically it has to be preformed manually by an expert or with the help of an expert. The major difficulty lies in the fact that every application might have its own specific rules

and requirements; and there might never be a universal method to construct a Bayesian network. Bayesian network experts have to have intensive communication and interaction with the domain expert. Some researchers investigated the techniques to construct efficient Bayesian networks automatically from data [45]. It starts with a simple initial network, evaluates the networks that result from every possible incremental change to the current network, and then applies the best of the changes before iterating. However, to develop the best algorithm for the changes could be challenging and elusive, which is one of the hottest topics in the KDD researches.

We will not adopt this incremental method for constructing our network, because the emphasis of this research should be put on the knowledge discovery, especially human-computer-interactive knowledge discovery, not the Bayesian network technology itself. Instead, the way we used to build Bayesian networks is to use knowledge engineering. A knowledge engineer can interact with an on-the-site domain expert to identify qualitative problem aspects, such as direct relationships between the problem variables, the adoption of certain variables and the withdrawal of irrelevant events. The relationship and variables adopted then become encoded in the network structure. In most cases, the domain expert can even provide conditional probability estimates to populate the associated distributions.

CHAPTER 3: KNOWLEDGE DISCOVERY IN DATABASES

3.1 Introduction to Knowledge Discovery in Databases

Knowledge discovery in databases (KDD) is the non-trivial extraction of implicit, previously unknown, and potentially useful information from data [20]. Here *data* are a set of facts (for example, cases in a database), and *pattern* is an expression in some language describing a subset of the data or a model applicable to the subset. Hence, in our usage here, extracting a pattern also designates fitting a model to data; finding structure from data; or in general, making any high-level description of a set of data. The term *process* implies that KDD comprises many steps, which involve data preparation, search for patterns, knowledge evaluations, and refinement, all repeated in multiple iterations. By *nontrivial*, we mean that some search or inference is involved; that is it is not a straightforward computation of predefined quantities like computing the average value of a set of numbers.

The basic problem addressed by the KDD process is to map low-level data, which are typically too voluminous to understand and digest easily, into other forms that might be more compact, more abstract, or more useful. Another term often associated with KDD is data mining. According to Fayyad [16], data mining is the application of specific algorithms for extracting patterns from data, while KDD refers to the overall process discovering useful knowledge from data. Therefore, data mining is actually a step in the KDD process consisting of applying data analysis and discovery algorithms that, under

acceptable computational efficiency limitations, produce a particular enumeration of patterns over the data.

KDD is an active research field, rapidly expanding with promising potentials for great applicability. The current interest in KDD can be stemmed from some widely reports of successful KDD applications in real world, e.g. in SKICAT, a system used by astronomers to perform image analysis [18]. However, the application could be much more extensive, including various areas like, business, manufacturing, telecommunication, engineering, military, etc. It is anticipated that commercial database systems of the future will include KDD capabilities in the form of intelligent database interfaces. Due to the potential applicability of knowledge discovery in so many diverse areas, there are growing research opportunities in this field.

3.2 A Framework for Knowledge Discovery in Databases

Figure 3-1 [17] depicts the basic components of our prototypical system for knowledge discovery in databases. At the core of the system is the discovery method, which computes and evaluates pattern on their way to becoming knowledge. The input to the discovery method include raw data from the database, information from the data dictionary, additional domain knowledge, and a set of user-defined biases that provide high-level focus. The output is discovered knowledge that can be directed to the user or back into the system as new domain knowledge. Because discovery systems vary considerably in their design, this model represents only a prototypical system. A discovery system need not include all these aspects. The feedback of discovered

knowledge into the store of domain knowledge, for example, is present in few existing system.

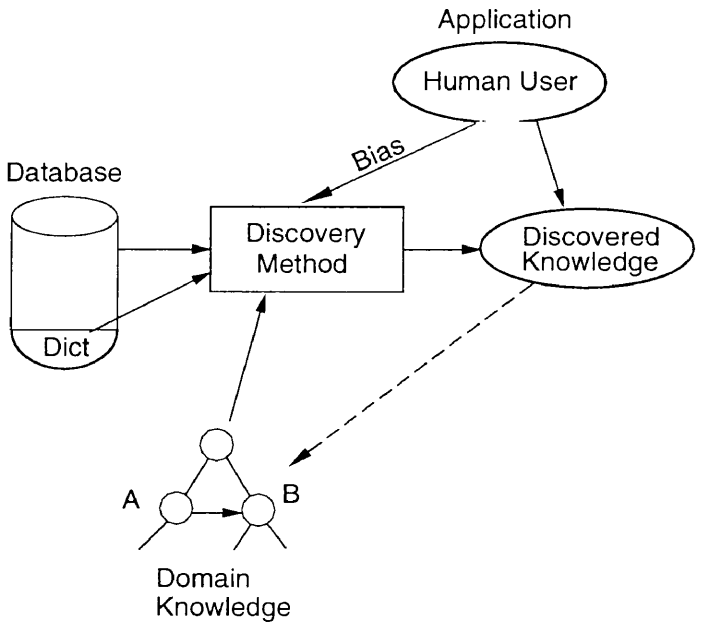


Figure 3-1. A prototypical framework for KDD

3.3 An Overview of the Steps That Compose the KDD Process

As shown in Figure 3-1, the core of a KDD system is the KDD process that generates and evaluates knowledge. To better understand a KDD process, we broadly outlined its basic steps in Figure 3-2. Note that the KDD process is interactive and iterative involving many decisions made by the user, we didn't show all the arrows that indicate that the loops can, and do occur between any two steps. Here we summarize the steps as follows:

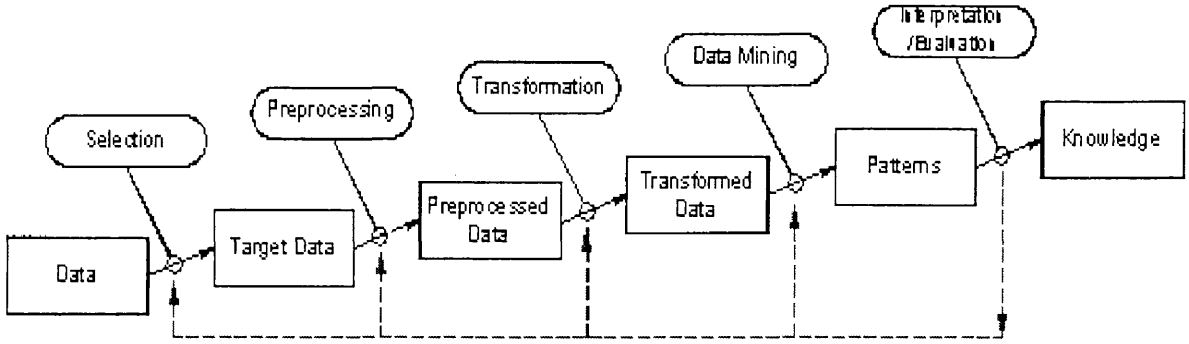


Figure 3-2. An overview of the basic steps that compose the KDD process

1. Developing an understanding of the application domain and the relevant prior knowledge and identifying the goal of the KDD process from the customer's viewpoint.
2. Creating a target data set: selecting a data set, or focusing on a subset of variables or data samples, on which discovery is to be performed.
3. Data cleaning and preprocessing. Basic operations include removing noise if appropriate, collecting the necessary information to model or account for noise, deciding on strategies for handling missing data fields, and accounting for time-sequence information and known changes.
4. Data reduction and projection: finding useful features to represent the data depending on the goal of the task. With dimensionality reduction or transformation methods, the effective number of variables under consideration can be reduced, or invariant representations for the data can be found.

5. Choosing the function of data mining: includes deciding the purpose of the model derived by the data-mining algorithm.
6. Choosing the data mining algorithm(s) and selecting method(s) to be used for searching for data patterns. This process includes deciding which models and parameters might be appropriate and matching a particular data mining method with the overall criteria of the KDD process (for example, the end user might be more interested in understanding the model than its predictive capabilities).
7. Data mining: searching for patterns of interest in a particular representational form or a set of such representations. The user can significantly aid the data-mining method by correctly performing the preceding steps.
8. Interpreting mined patterns, possibly returning to any of steps 1 through 7 for further iteration. This step can also involve visualization of the extracted patterns and models or visualization of the data given the extracted models.
9. Acting on the discovered knowledge: using the knowledge directly, incorporating the knowledge into another system for further action, or simply documenting it and reporting it to interested parties. This process also includes checking for and resolving potential conflicts with previously believed (or extracted) knowledge.

The data-mining component in step 7, by far, has received the most attention in the literatures. Data mining involves fitting models to or determining patterns from observed data. Deciding whether or not the models reflect useful knowledge is a part of the overall interactive KDD process for which subjective human judgment is usually required. A wide variety and number of data mining algorithms are described in the

literature — from the fields of statistics, pattern recognition, machine learning, and databases. Our research focused on developing an interactive mining system, with the assumption that the target data is well chosen and accurate. However, the other steps are as important for the successful application of KDD in practice.

3.4 Aiming at Future Directions of KDD

Knowledge discovery is a promising and challenging field. Some aspects of discovery in databases, such as finding simple formulas to fit scientific data or inducing decision trees for classification, are relatively well understood, many more aspects are in need of research. Frawley predicted some of the research directions in 1992, some especially promising elements include [20]:

Incorporation of background and associated knowledge. Because discovery is computationally expensive, additional knowledge regarding the form and content of data, the domain described by the database, the context of a particular discovery episode, and the purposes being served by discovery, are often used to guide and constrain the search for interesting knowledge. We refer to this form of information as domain knowledge or background knowledge. Domain knowledge assists discovery by providing high-level search focus. In any discipline, from scientific discovery to business strategy, there is often a limit to the acquisition of knowledge from a single source of raw data, whatever the technique employed. Incorporating any associated knowledge significantly increases the efficiency of the process and the quality of the knowledge discovered.

More comprehensive types of data. Most knowledge discovery techniques today deal with relatively simple forms of target data. Dealing with more comprehensive, non-oversimplified, real-world types of data, such as ultrasound images and mixed data types, is a key direction for the future.

Human-computer interaction. Any human and any computer each reflect individual strengths and weaknesses. For example, a computer is incredibly fast and unbiased but lacks common sense and intuition. Interactive systems will provide, perhaps, the best opportunity for discovery in the near term. In such systems, a knowledge analyst is included in the discovery loop. This approach combines the best features of human and machine: Use human judgment but rely on the machine to do search and to crunch numbers. The interactive approach requires the discovered knowledge to be presented in a human-oriented form, whether as written reports or visual and sound patterns. Such novel output presentations might allow the use of the phenomenal perceptual capabilities of humans. Tools need to be built to support effective interaction between the user and the discovery system. Also algorithms need to be reexamined from the viewpoint of human-oriented presentation.

Hybrid systems. Each KDD technique has advantages and weaknesses. Hybrid systems will help compensate for a particular system's weaknesses, thus creating new approaches to knowledge discovery.

Our research explores an interactive and iterative KDD method that tackles two of the problems represented above: the incorporation of the background knowledge and the human-computer interaction. As we discussed in section 1.3, easy background

incorporation is one of the reasons we chose a Bayesian network as the representation model. It has the advantage of allowing prior knowledge about the domain and data to be included in a relatively easy and natural framework.

One of the other major advantages of a Bayesian networks lies in the fact that it can be represented explicitly using a graph. Communication is a bilateral process. An efficient communication requires the two sides to represent their ideas in a way that can be perceived easily and effectively by the other communicator. Therefore, representing the information in a precise manner is crucial. A graph offers a desirable way of representing information. Even a quick glance of a graph can catch quite a lot of information: nodes tell the events that involves in the problem, edges the relationships among the events. Furthermore, The probability distribution of a Bayesian network provides quantitative information regarding the problem issue. On the other hand, a Bayesian network also eases the computer's understanding of its communicator, the human being: The structure of a network incorporates the user's comprehension of the problem in a very naturally way, and modifications therefore can be made directly to the network without making any changes to the source data. As soon as the modifications are made, they take effects right away. No intermediate processing is needed.

In all, the feature of graphical representation makes Bayesian networks a potential and promising tool of interactive knowledge discovery. In our research, we will exploit its graphical capability to facilitate the process of knowledge discovery, and to create a dynamic human-computer interaction, which in return provides a viable mechanism to control the refinement and the growth of the Bayesian network.

CHAPTER 4: A GOAL-DRIVEN INTERACTIVE AND ITERATIVE KDD MODEL

4.1 Network Structure

In this chapter, we will represent a computational model to apply Bayesian networks to knowledge discovery under uncertainty in decision support system. Major features of this model include user-computer interaction and iterative information extraction. The user plays a primary role when determine the acceptance or refusal of discovered information, while the computer, as a supporting roles, handles all the computation.

The process of knowledge discovery in this model starts from the construction of an initial Bayesian network. The initial Bayesian network is constructed to form and represent the initial set of goal and sub-goals. These goal and sub-goals are to be expanded in the database exploration process so that the leaf nodes hold quantitative information derived from the source data. This enables the knowledge discovery process preserve a content-controllable manner. Major components of the initial Bayesian Network are:

- An ultimate goal, which is the objective of the information retrieval process and serves as the starting point of the Bayesian network construction (the root if the causal network has a tree-like structure).

- A set of variables (will be regarded as sub-goals) with certain relations with the ultimate goal. Here the user or the domain expert plugs in his or her own knowledge and expectations.

An initial Bayesian network could be rather coarse in terms of information content it carries. Without further refinement, it is still disjointed with the variables of relevant objects and their probabilities obtained from source data. To set up a connection, a network decomposition process is invoked. Our purpose of this process is to itemize the complexity involved in the dependency representations for the goal and sub-goals. In the decomposition process, the node connections in the Bayesian network relevant to the source data are identified. Intermediate nodes and links are added to the Bayesian network to set up the path from the goal and sub-goals to the leaf nodes where source data detection can be carried out. Since the network decomposition is essentially coherent with and part of the knowledge discovery process, we will discuss it later together with the knowledge discovery procedure.

After refinement, a tentative Bayesian network that can carry out a propagation consists of the following items: (a) a root node that contains the ultimate goal for the knowledge discovery; (b) some leaf nodes that serves as the interface to the source data; and finally (c) some intermediate or sub-goal nodes in-between the root and the leaves. There may be more than one root in the network based on the need of different application. However, one of the roots, as we pointed out, must contain the ultimate goal. Figure 4-1 gives a conceptual picture for a tentative Bayesian network structure. As you

can see from Figure 4-1, the intermediate nodes function as a bridge where the connection between the customer desire and the source data is established.

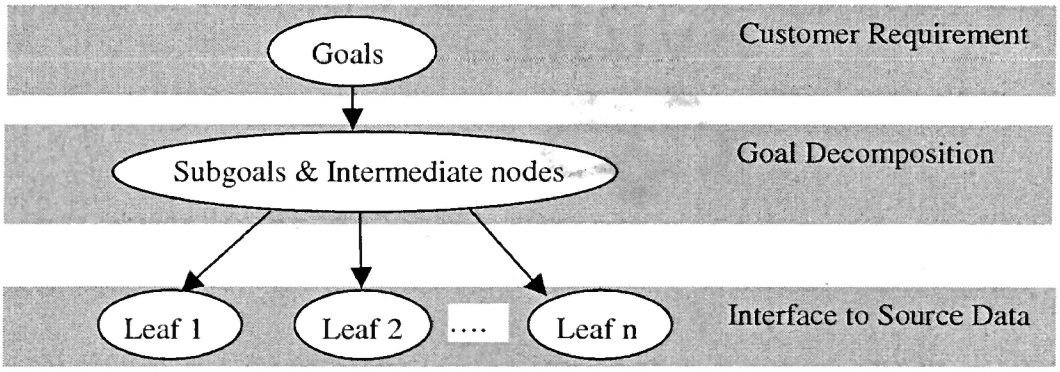


Figure 4-1. The feasible network structure for our application

4.2 A Goal-Driven Interactive and Iterative KDD Model

When we have a network with the conceptual structure shown in Figure 4-1, we can now start the KDD process. The KDD process will not only extract information related to the source data, but also refine the existing network. Remember we said the Bayesian network is a good information container. So eventually, the acquired knowledge will consist of two parts: (1) the Bayesian network specifically developed to model the problem application, and (2) the quantitative information retrieved from source data by applying the Bayesian discovery technique.

A conceptual model has been developed to represent the actual user-guided knowledge discovery process. This model employs an incrementally constructed Bayesian network with user-directed information. Figure 4-2 depicts the processing loop and main functional units of this model. Two streams are involved in the model:

1. Top-down stream: the user enters the expectation value for the goal in a tentative Bayesian network, propagation will then be carried out to calculate the expected values for all the leave nodes.
2. Bottom-up stream: the user instantiate certain variables in the network, propagation will then be carried out to see how much the instantiation will affect the goal and all other nodes.

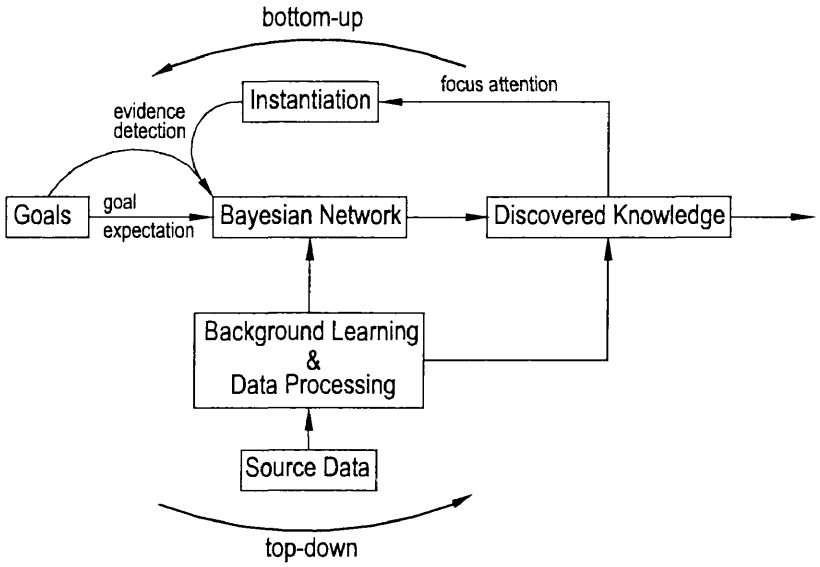


Figure 4-2. A user-guided goal-driven interactive and iterative KDD model

The first stream calculates the expected values for all the nodes under a certain requirement given to the goal node. For instance, if we want the goal event, which is the root node (if the network has a tree structure) or one of the root nodes, to happen 80 percent of time, we then specify the probability of the goal node to be (0.8, 0.2) and carry out a top-down computation. By doing so, we will get the probability values for all the

nodes in the network. We can consider these values as required criteria in order to ensure the probability of goal realization. The second stream handles the other way. If we have a certain case and we want to use our system to test how good this case would be. Then we input all the information related to this case to the network and commit a bottom-up computation, from which the goal node will get a new probability value. For example, if the computation gets a result that, under a specific case, the goal node receives a probability of (0.6, 0.4), which means that 60 percent of time the event related to goal node will be true. And suppose the expected value for the goal is 80 percent, then we can conclude that the case being studied may not be a successful one.

Both streams can continuously be managed and combined. Figure 4-3 outlines the entire KDD process.

```
Do
{
        Step 1: Build or modify the network representing the query
        Step 2: Generate information
                2.1 Enter needed background information
                2.2 Calculate the probability values under given expectation
                2.3 Instantiate certain features in the Bayesian network
                2.4 Calculate the posterior probability of relevance
        Step 3: Analyze derived information
} while(needed)
```

Figure 4-3. An outline of knowledge discovery process using the proposed model

In Figure 4-3, step 1 is a traditional network construction and modification step. Network nodes and edges are added or removed here. Before go over to step 2, the network should have a structure as shown in Figure 4-1. Step 2 contains the main propagation computation. The user enters background information, like conditional probabilities, in step 2.1. The user enters an expectation value for the goals and carry out step 2.2, which implements the top-down computation. This computation calculates the probability values for all other nodes in the network. Upon evaluating the probability values, the user may reject certain variables or may modify the network connection according to his/her understanding. Furthermore, the user may want to focus his/her attention on certain aspects of the information by instantiate some of the variables in the network. The instantiation then starts the bottom-up computation contained in step 2.3. Under the given evidences on some of the variables, the goal node will receive a new probability value. In step 3, the user speculates further into the relevance represented by the network by comparing the new value to the old expectation. More modifications to the network may be made to better serve the knowledge discovery process and also better match the domain problem. This whole process will repeat as many times as needed. The ending of the process is largely determined by the imagination or the cautiousness of the user. When the user feels that no further network refinement is needed or that meaningful information has already been retrieved, the KDD process ends up.

In the above process, we observe two major features of our technique: (1) Interaction: We established an easy communication between the computer and the user

and they work together to support the decision-making. (2) Iteration: The process is a loopy refinement and testing process.

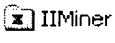
CHAPTER 5: EXPLANATION OF THE SOFTWARE PROTOTYPE

5.1 User Menu of the Software Prototype

In chapter 4, we explained the new computational model we developed for a KDD process. The development of this model is the first part of the proposed task in the thesis project. The second part, consequently, involves the design and development of a software prototype for implementing the computation model. The software prototype will also serve as a platform for the test and evaluation of the computational model for practical applications.

In this chapter, we first give out a user menu to explain the steps of using the software prototype we designed. Following that, we will use an example to reiterate the usage of the software prototype and the computational model.

5.1.1 Unpacking

1. Open Window Explorer, go to zip disk or CD, find the zip file named *IIMiner*. The icon for this file is: 

2. Put the mouse on the file icon and right click. On the pop-up menu, choose *Extract here*, as shown in Figure 5-1, to unzip. Unzip will create all the files associated with the program.

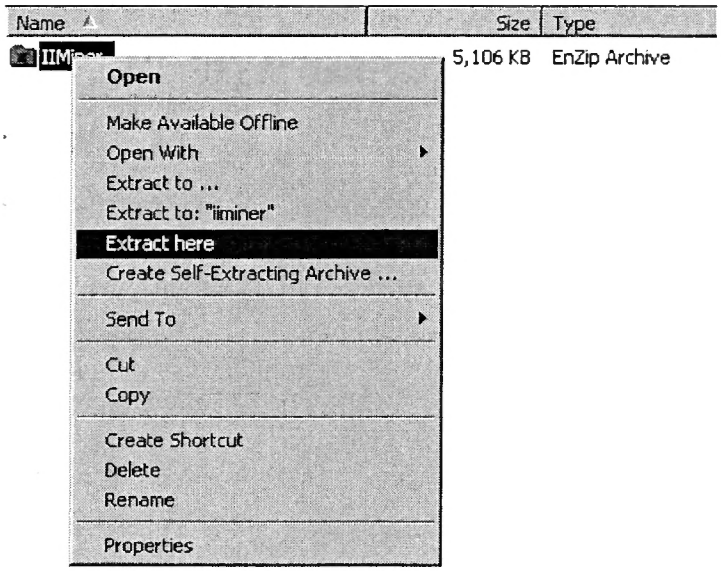


Figure 5-1. Unzip the software prototype

3. To run the program, double click on file icon IIMiner . The first window you will see after you launched the program is the *Data Come From* dialog box as shown in Figure 5-2. This window asks for the location of the source data. Two options has been offered in this window:

   - Data Warehouse: Source data are stored in a remote data warehouse. This option enables the user to connect to the data warehouse, and access the data from the data warehouse. For this option, we need ODBC to connect to the host database. Another student, Miao Chen, has done the implementation to this option. To get more information about his work, please refer to [9].
   - Local Data: The source data are located on the local machine and stored in files. This option does not require any on-line connection, and has been our focus of implementation.

To continue, click on *Local Data* button.

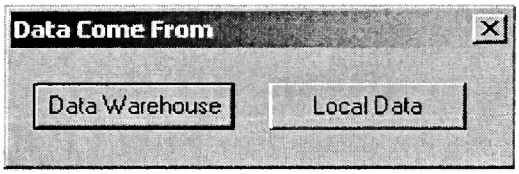


Figure 5-2. The *Data Come From* dialog

4. The click on *Local Data* will trigger the interface mainframe as shown in Figure 5-3. We will explain the function buttons later in section 5.1.2. The three information areas contained in the mainframe are:

   - File List Area: List all available files related to the KDD process
   - Source Data List Area: Display the information contained in a specified source file. To specify one file, move the mouse over to one of the files listed in the *File List Area* and single click. The color of the file name will reverse to represent the selection.
   - Bayesian Network Representation Area: Display a Bayesian network

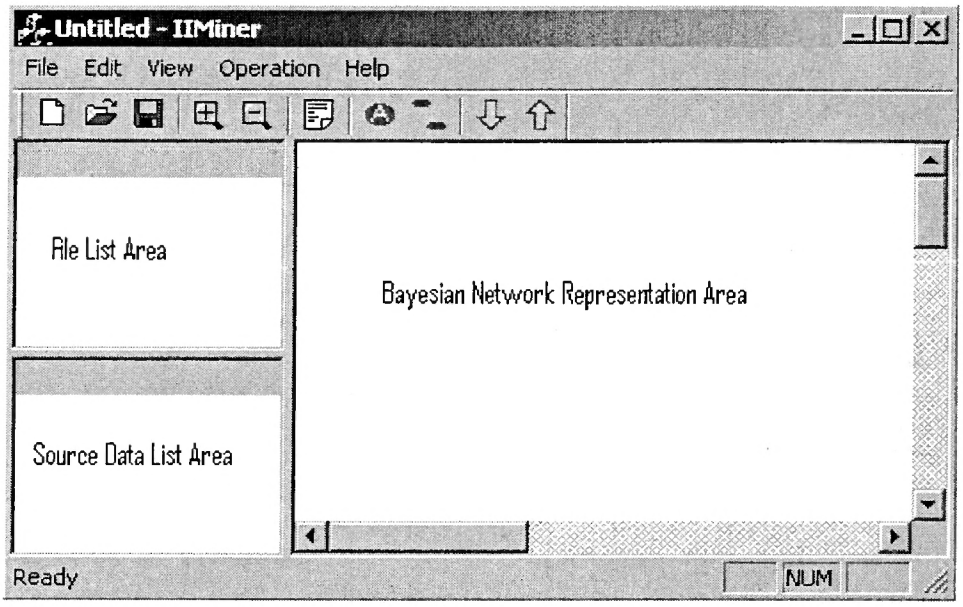


Figure 5-3. The main window for the software prototype

5.1.2 Understanding Tool Bar Buttons

Figure 5-4 shows all the toolbar buttons. Below we explain their functionalities one by one as numbered.

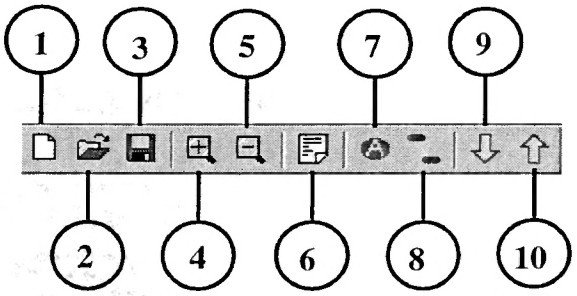


Figure 5-4. The tool bar buttons

1. New File

Start a new file. Here *file* refers to a bitmap file that contains all the information related to a Bayesian network: nodes, edges, probability distributions, and λ and π values. When you open an existing file, you will see exactly the same file as the one you last edited and saved.

2. Open File

To open an existing file, click on *Open File*, the *File Open* Dialog appears as shown in Figure 5-5. Find the file you want to open and click on *Open* Button.

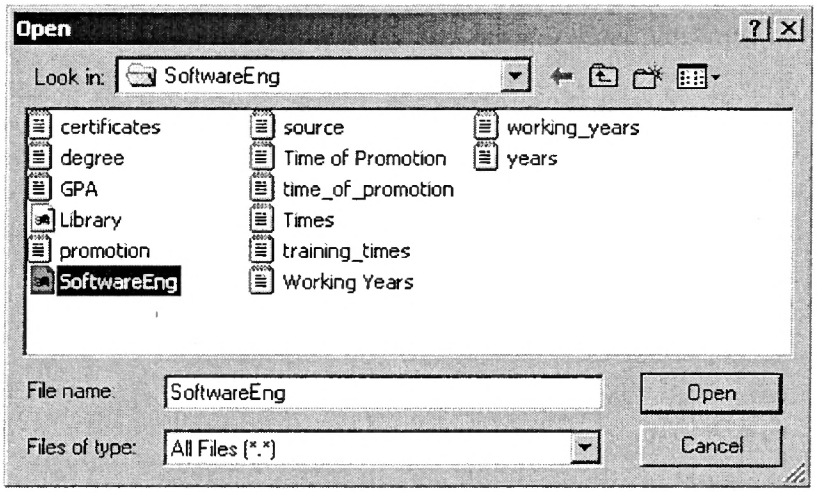


Figure 5-5. The *Open File* dialog

3. Save File

Click on this button will save all the information related to the working Bayesian network. A *File Save* dialog, which looks almost the same as in Figure 5-5, will help you to specify the path and fold you want to save the network. You will have to type a file name in the *File Name* text box if the network has never been saved before or if you want

to save the network to another file. Click on *Save* button (instead of *Open* button in Figure 5-5) to save a network.

4. Zoom in

Click to enlarge the window display area. When you move the nodes or edges of the Bayesian network that you are building, part of the network may go beyond the display area. Use this button to enlarge the window and regain the control of the out-of-sight part.

5. Zoom Out

Click to shrink the window display area.

6. Create Data

Click to create artificial data. For the purpose of demonstration, when there is no real world data available, this button creates the needed data. When clicked, a *Create Data* dialog will ask you to input to a minimum, a maximum, and a unit for each leaf node in the network. These three values together will determine all the possible values for a node. For instance, Figure 5-6 shows how we are going to decide the possible values for node *Certificate*. If we input 0 as the minimum, 10 the maximum and 1 the unit, totally there would be 11 possible values for *Certificate*, and these 11 values are the numbers from 0 to 10. If the node has a numerical data type, that is all we need to do. When the node takes non-numerical values as its data, a little more work has to be done. Now we still need the minimum, maximum and unit values, but they do not bear any real meaning but to determine the number of possible non-numerical values. After we input the three values, we then need to choose the *Non-Numerical* check box. If you look at

Figure 5-6, you may notice the *Specify...* button is actually disabled. Click on the *Non-Numerical* check box will activate the *Specify...* button. Once the button is activated, we can click on it to get a dialog window ask you to input the corresponding alphabetic values as shown in Figure 5-7.

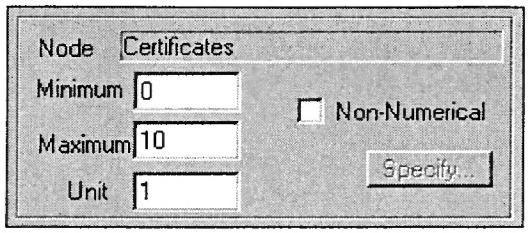


Figure 5-6. Information input for a node in *Create Data* dialog

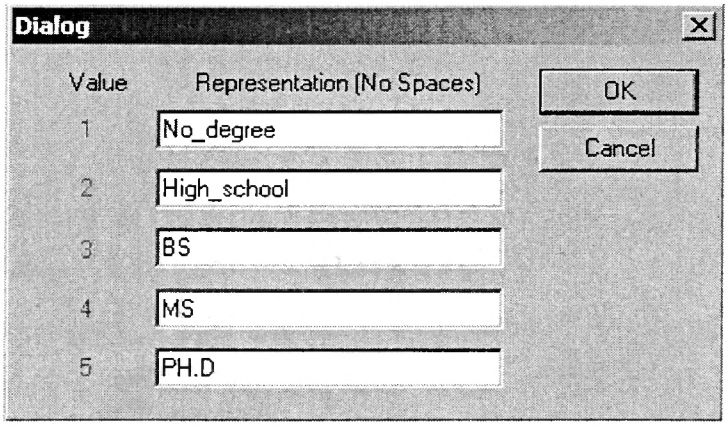


Figure 5-7. An example for non-numerical data value input

After we have input all the information we need to decide the values for all the leaf nodes, push the *Create Data* button on the *Create Data* dialog window. Five thousand randomly selected results will be created for each leaf node.

7. Create Node

When click, the *Insert Node* dialog appears as showed in Figure 5-8.

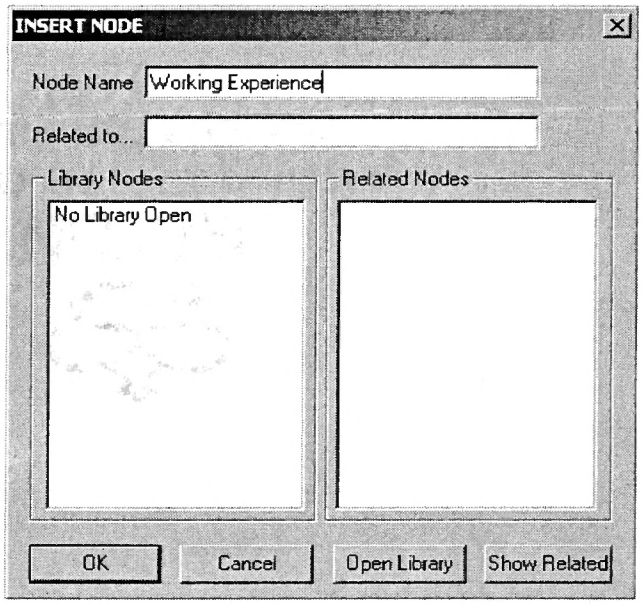


Figure 5-8. The *Insert Node* dialog

There are three ways to insert a node to the current Bayesian network:

One, type a node name directly in the *Node Name* text box. As shown in Figure 5-8, the user intends to add a new node, *working experience*, to the current network.

Two, choose from a library node. By saying *library*, we mean a given graph that is associated with the problem background. The library graph certainly contains a great deal of information, of which some may be distractive or irrelevant. However, it also contain useful information that may serve as good guidelines for the domain expert. If there is such a library available, then click on *Open Library* button, the *File Open* Dialog, as shown in Figure 5-5, appears. Choose the library you want to open and then click the *Open* button. All the nodes contained in the library graph will be listed in the *Library*

Nodes list box. Figure 5-9 shows one example when a library is open. If the user wants to add one of the library nodes to the current Bayesian network, just click on that specific node. *Node Name* text box will automatically display the chosen node. In Figure 5-9, when the user selects a library node, *working experience*, this node also appears in the *Node Name* text box. If *OK* button was pressed when exiting the *Insert Node* dialog, then the chosen library node will be inserted to the current network.

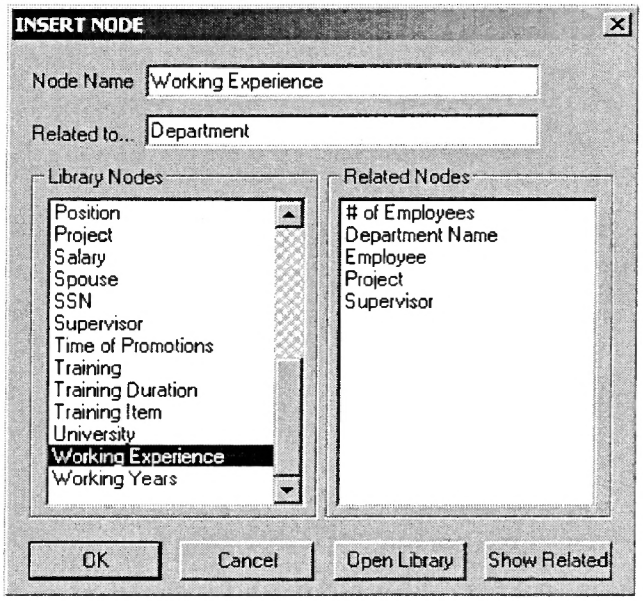


Figure 5-9. The *Insert Node* dialog with a library open

The last way to add a node is to select one from the *Related Nodes* list box. If you want to know the relationship among the library nodes, type a node name in *Related to* text box, and click *Show Related* button. All those library nodes, which are adjacent to, or in other words, have connection with the node shown in *Related to* text box, will be displayed in *Related Nodes* list box. In Figure 5-9, the *Related Nodes* list box shows all

nodes related to the node *Department*. Just as we did above by choosing a library node as the new node we want to add, we can also choose a node from the *Related Nodes* list box.

8. Create Link

To create a new link, first click on the *Create Link* tool bar button, then select two nodes by single clicking them. For the two selected nodes, three causal relations are provided as shown in Figure 5-10. Choose one of them and then Click *OK*.

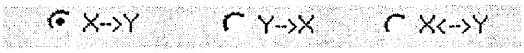


Figure 5-10. The three causal relations provided to connect two nodes

9. Top-down Calculation

Given the probability of the root nodes, calculate the probability values of other nodes.

10. Bottom-up Calculation

Given the evidence of some nodes, calculate the probability of the root nodes.

5.1.3 Mouse Activities

The proposed software prototype provides the following three mouse activities:

(1) Data input and detail information check by mouse double clicks. To input or change the conditional probability values, double click on the link to trigger a *Link State* dialog. Figure 5-11 shows the information associated with link "*Software Engineer* →

Qualification": λ and π message values on the link and the conditional probabilities. The inputs of the conditional probabilities are also done here.

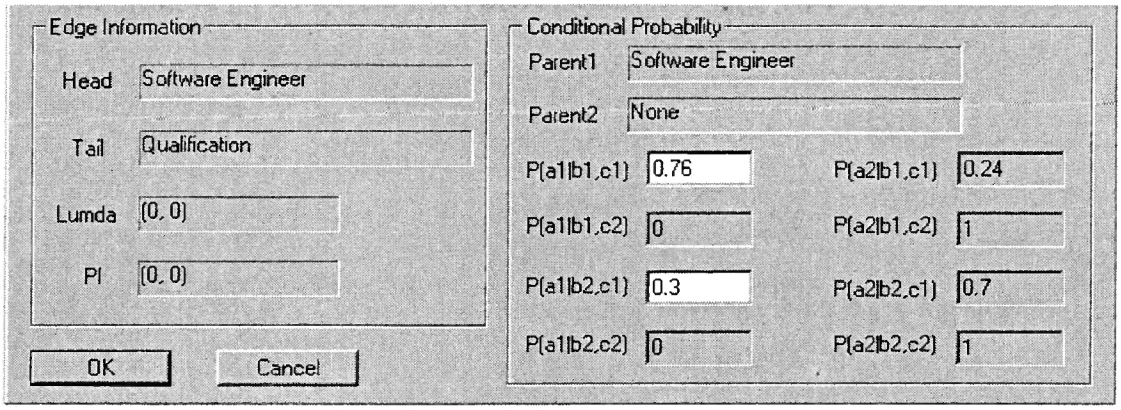


Figure 5-11. The *Link State* dialog

To input or change the probability value for a node, double click on the node opens a *Node State* dialog, as shown in Figure 5-12, which displays the information, λ, π and probability values, of the *Software Engineer* node. Since *Software Engineer* node is a root node, you can input the expected probability value in the *Node State* Dialog. When the node is a non-root node, however, the probability values are read-only since there is no probability input needed. We can instantiate any node by choose the *instantiate* P_1 or the *instantiate* P_2 radio button in the dialog. By choosing one of them, we will tell the computer that one of the node events happens while the other one doesn't. For example, if we choose *instantiate* P_1, we actually change the node's probability to be (1, 0). Since the first event of a node is a true event and the second a false one, instantiating P_1 therefore creates a situation in which the chosen event for sure happens.

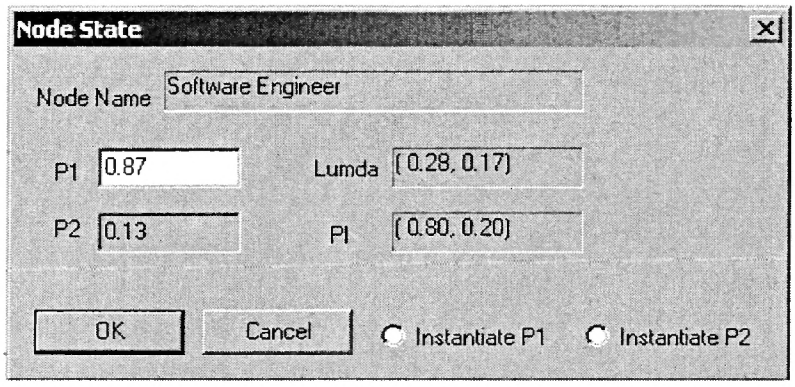


Figure 5-12. The *Node State* dialog

(2) Quick information check by docking the mouse position. In addition to checking the information by opening *Node State* or *Link State* dialogs, we can also have a quick glimpse of some of the node or link information by just moving the mouse over. For instance, if you move the mouse over a certain node, the software automatically displays the probability values of the node as shown in Figure 5-13. Figure 5-14 shows the information displayed by moving a mouse over an edge.

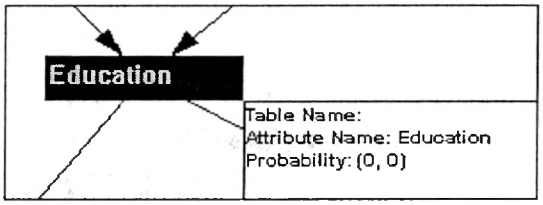


Figure 5-13. Information displayed when the mouse stops on a node

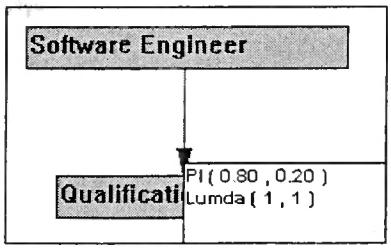


Figure 5-14. Information displayed when the mouse stops on an edge

(3) Node or link removal assisted by docking the mouse position. To remove a node or a link, move the mouse over to the desired node or link (the color of the node or link reverses when the mouse is over it), and press the *Delete* key on the keyboard.

5.2 Exemplar Explanation of the Software Prototype

In this section, we consider a decision support system where the user is interested in the discovery of some common features of qualified candidates for a software engineering position. The source data given to the user consists of employment information for previous qualified software engineers including their educational background, working history and other related information. One can carry out the goal-driven process in the following manner. Let *Software Engineer* be a goal for a knowledge discovery process, whose purpose is to find out the requirement under which a candidate could be a good match for the goal or if given a candidate's information, how qualified this candidate is for such a position. An initial network that consists of the goal and the sub-goals for this application is given in Figure 5-15.

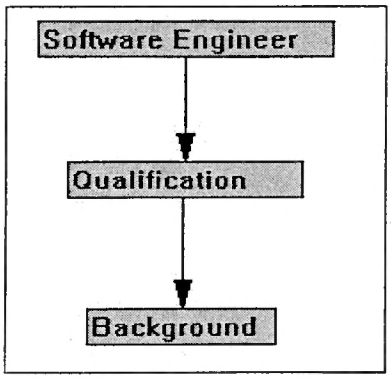


Figure 5-15. The initial network of the *Software Engineer* demonstration

We look at the source data related to the sub-goals: *background.* Since the source data contains information about the employer's education and working background, we then add two more node, *education* and *working experience* to the network. Keeping expanding the initial network by considering the source data contents at the meantime, we obtain the network in Figure 5-16.

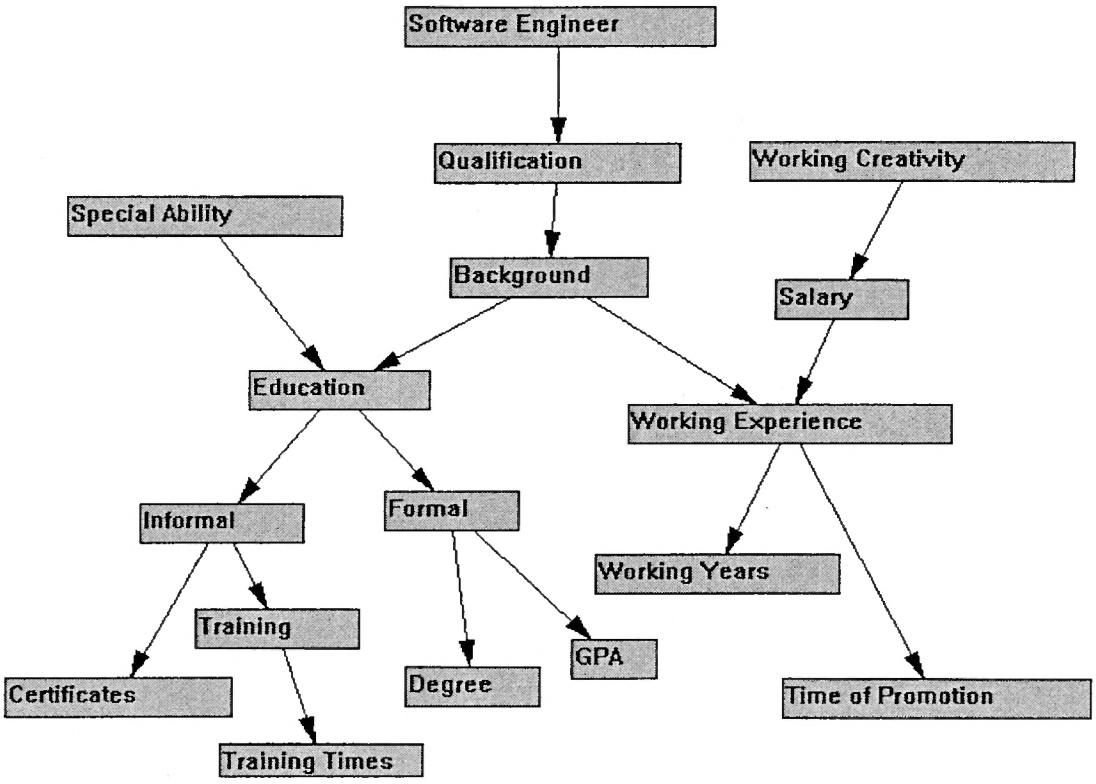


Figure 5-16. The Bayesian network structure of the *Software Engineer* demonstration

As we can see from Figure 5-16, the ultimate goal for this application is contained in one of the root nodes: *Software Engineer.* The leaf nodes are: *Certificates*, T*raining*

Times, *Degree*, *GPA*, *Time of Promotion*, *Working Years*. Click on the *Create Data* toolbar button to create artificial data for this demonstration. Figure 5-17 shows the mainframe after we generated the source data. When there is no source data available, the *File List* area and the *Source Data List* area are empty. Now the *File List* area lists the names of the source files. When one of the files is selected, the *Source Data List* area displays the content of that file. As shown in the graph, the *Source Data List* area shows the source data contained in "*Degree.txt*".

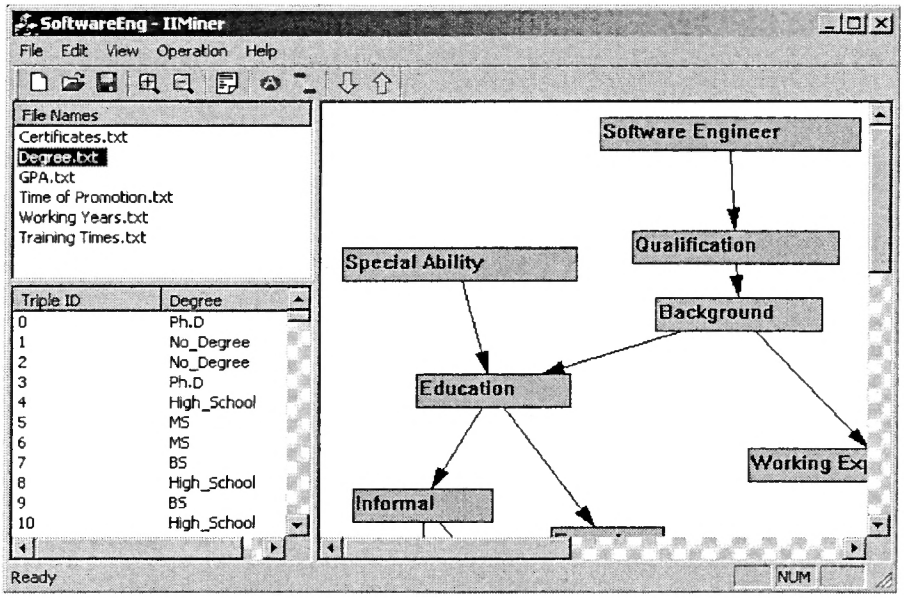


Figure 5-17. The mainframe with available source data for the *Software Engineer* example

The user now enters the associated conditional probability values, input an expectation probability value for all the root nodes, and do a top-down propagation. Suppose we expect that a potentially good software engineer should make the goal *Software Engineer* happens at least 80 percent of time. Therefore, the expectation

probability values for the goal becomes (0.8, 0.2). We double click on the *Software Engineer* node and key in the information as shown in Figure 5-18. Continue input the expectation values for all the root nodes.

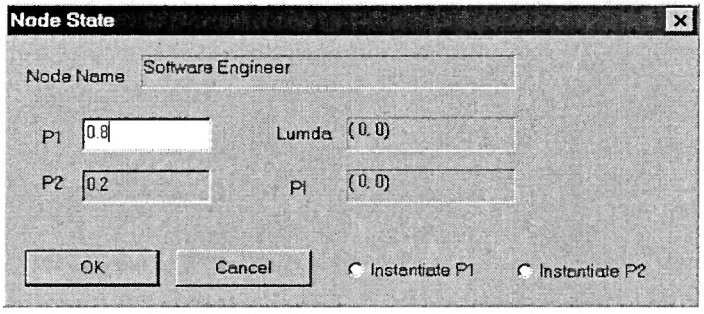


Figure 5-18. Goal expectation input in the *Software Engineer* example

Now with the given expectation probabilities, if we press the top-down computation toolbar button, we will get all the required values for each of leave node as one shown in Figure 5-19. When examining these values, the user may determine to remove the node *Training Times* since it does not appear to be crucial for the goal. Or the user may also want to step further to test how much the *GPA* will effect the qualification. Then he instantiates P_2 of the *GPA* node, which means the *GPA* is bad, and starts a bottom up propagation. Suppose the probability value for the goal changed from 0.8 to 0.78 due to this instantiation. According to the user's understanding upon the problem issue, he may think the discrepancy is negligible and the GPA node doesn't have too much affect on the goal. Therefore, he may want to remove this node from the network. Continue testing other nodes or adding more nodes to the network and more

iterations of the process will be committed to get a final network that suits for the problem. The user will eventually get a network that suits fro this specific problem.

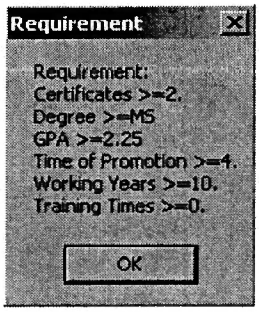


Figure 5-19. The discovered knowledge after one top-down computation

CHAPTER 6: CONCLUSION

6.1 Conclusion

In this thesis, we have explored a new KDD methodology based on a model developed for an interactive and iterative KDD process. We have also designed a software prototype to testify and implement this methodology. Later we demonstrated the use of the software prototype by exploring an application example. Since this KDD approach we studied is interactive, the user can always provide his or her intention to guide the search in a timely fashion. Consequently, the scope of directions to consider can be reduced, unwanted knowledge patterns can be excluded, and the overabundance problem can be remedied. When the discovery process iterates, unwanted or unrelated information are filtered out and the knowledge pattern shrinks to be more realistic. This new approach is most suitable to the cases of decision support where the user knows what he/she wants and is able to refine his/her goal step by step, that is, a kind of control of the knowledge discovery process is desirable to the user.

6.2 Suggestions

Further research is suggested in the following areas:

1. The incorporation of the discovered knowledge to the decision support system;
2. Use real data from the real world to test the software prototype;

3. The combination of the system with other technology.

REFERENCE

1. Ahn, Jae-Hyeon; Ezawa, Kazuo J., "Decision support for real-time telemarketing operations through Bayesian network learning," *Decision Support Systems*, Vol. 21, No. 1, pp.17-27, Sep. 1997
2. Ambrosio, B. D., "Inference in Bayesian Networks," *AI Magazine,* Vol. 20, No.2, pp. 21-35, 1999
3. Becker, B.G., "Using MineSet for Knowledge Discovery," *IEEE Computer Graphics and Applications*, Vol. 17, No. 4, pp. 75-79, Jul/Aug 97
4. Breese, J. S.; Goldman, R. P.; Wellman, M. P., " Introduction to the Special Section on Knowledge-Based Construction of Probabilistic and Decision Models," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 24, No. 11, pp. 1577-1579, 1994
5. Campbell, M., "Knowledge Discovery in Deep Blue," *Communication of the ACM*, Vol. 42, No. 11, pp.65-67, 1999
6. Carter, C. L.; Hamilton, H. J., "Efficient Attribute-Oriented Generalization for Knowledge Discovery from Large Databases," *IEEE Transactions on Knowledge and Data Engineering,* Vol. 10, No. 2, pp. 193-207, 1998
7. Charniak, E., "Bayesian Networks without Tears," *AI Magazine*, Vol.12, pp. 50-63, Winter 1991
8. Chavez, R. M.; Cooper, G. F., "A Randomized Approximation Algorithm for Probabilistic Inference on Bayesian Belief Networks," *Networks*, Vol. 20, pp. 661-685, 1990

9. Chen, M., "IIMiner: An Informed Interactive Query Approach for Knowledge Discovery from Heterogeneous Data Sources," University of Nebraska at Omaha, Jul. 2000

10. Chung, H. M.; Gray, P., "Special Section: Data Mining," *Journal of Management Information Systems*, Vol. 16, No. 1, pp. 11-17, Summer 1999

11. Cook, D. J.; Holder, L. B., "Graph-Based Data Mining," *IEEE Intelligent Systems & Their Applications*, Vol. 15, No. 2, pp.32-39, 2000

12. Cooper, G., "Computational complexity of probabilistic inference using Bayesian belief networks," *Artificial Intelligence*, Vol.42, pp. 393-405, 1990

13. Darwiche, A.; Pearl, J., "Symbolic Causal Networks," *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI),* pp. 238-244, 1994

14. De Jong, A. K., "Evolutionary Computation for Discovery," *Communication of the ACM*, Vol. 42, No. 11, pp.51-53, 1999

15. Ezawa, K. J.; Norton, S. W., "Constructing Bayesian Networks to Predict Uncollectible Telecommunications Accounts," *IEEE Expert*, Vol. 11, pp. 45-51, Oct. 1996

16. Fayyad, M. U., "Data Mining and Knowledge Discovery: Making Sense out of Data," *IEEE Expert Intelligence Systems & Their Applications*, Vol. 11, No. 5, pp. 20-26, Oct. 1996

17. Fayyad, U.; Piatetsky-Shapiro, G.; Smyth P., "From Data Mining to Knowledge Discovery in Databases," *AI Magazine*, Vol. 17, No. 3, pp.37-54, Fall 1996

18. Fayyad, U.; Piatetsky-Shapiro, G.; Smyth, P., "The KDD Process for Extracting Useful Knowledge from Volumes of Data," *Communications of the ACM*, Vol. 39, No. 11, pp. 27-34, Nov. 1996

19. Fenton, N.; Neil, M., "Making Decisions: Using Bayesian Nets and MCDA," to be appeared on *Knowledge-Based Systems, 2001*, (available on http://www.dc.qmw.ac.uk/~norman)

20. Frawley, W. J.; Piatetsky-Shapiro, G.; Matheus, C. J., "Knowledge Discovery in Databases: An Overview," *AI Magazine*, Vol. 13, No. 3, pp. 57-70, Fall 1992

21. Fu, L. M., "Knowledge Discovery Based on Neural Networks," *Communication of the ACM*, Vol. 42, No. 11, pp. 47-50, 1999

22. Fung, R.; Favero, B. D., "Applying Bayesian Networks to Information Retrieval," *Communication of the ACM*, Vol. 38, No. 3, pp. 42-48, 1995

23. Glymour, C.; Madigan, D.; Pregibon, D.; Smyth, P., "Statistical Inference and Data Mining," *Communications of the ACM*, Vol. 39, No. 11, pp. 35-41, Nov. 1996

24. Goldman, R.; Charniak, E, "A Language for Construction of Belief Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 3, pp. 196-208, 1993

25. Han J. W.; Cai, Y. D.; Cercone, N., "Knowledge Discovery in Databases: An Attribute-Oriented Approach," *Proceeding of the 18th VLDB Conference*, Vancouver, B.C., Canada, 1992

26. Hand, D., "Data Mining: Statistics or More?" *The Amercian Statistician*, Vol. 52, No. 2, pp. 112-118, 1998

27. Heckerman, D., "Bayesian Networks for Data Mining," *Data Mining and Knowledge Discovery*, Vol. 1, No. 1, pp. 79-119, 1997

28. Heckerman, D.; Breese, J. S.; Rommelse, K., "Decision-Theoretic Troubleshooting," *Communication of the ACM*, Vol. 38, No. 3, pp. 49-57, 1995

29. Heckerman, D.; Meek, C.; Cooper, G., "A Bayesian Approach to Causal Discovery," *Microsoft Research*, Feb, 1997 (available on http://research.microsoft.com/~heckerman)

30. Heckerman, D.; Wellman, M. P., "Real-World Application of Bayesian Networks," *Communication of the ACM*, Vol. 38, No. 3, pp. 25-41, 1995

31. Hsu, C. N.; Knoblock, C. A., "Using Inductive Learning To Generate Rules For Semantic Query Optimization," *Advances in Knowledge Discovery and Data Mining*, eds. Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P.; and Uthurusamy, R., AAAI Press/The MIT Press, Menlo Park, CA., pp. 425-445, 1996

32. Imieliniski, T.; Mannila, H., "A Database Perspective on Knowledge Discovery," *Communications of the ACM*, Vol. 39, No. 11, pp. 59-64, Nov. 1996

33. Indrawan, M.; Ghazfan, D.; Srinivasan, B., "Using Bayesian Network as Retrieval Engines," *Proceedings 15th Text Retrieval Conference*, pp. 437-443, 1996

34. Lam, W., "Bayesian Network Refinement Via Machine Learning Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 3, pp. 240-251, 1998

35. Lavington, S.; Newburst, N.; Wilkins, E.; Freitas, A., "Interfacing Knowledge Discovery Algorithms to Large Database Management Systems," *Information and Software Technology*, Vol. 41, No. 9, pp. 695-617, June 1999

36. Li, Z. Y.; Ambrosio, B. D., "A Framework for Ordering Composite Beliefs in Belief Networks," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 25, No. 2, pp. 243-255, 1995

37. Mitchell, T. M., "Machine Learning and Data Mining," *Communication of the ACM*, Vol. 42, No. 11, pp. 30-36, 1999

38. Muggleton, S., "Scientific Knowledge Discovery Using Inductive Logic Programming," *Communication of the ACM*, Vol. 42, No. 11, pp. 43-46, 1999

39. Munakata, T., "Knowledge Discovery," *Communication of the ACM*, Vol. 42, No. 11, pp.27-29, 1999

40. Neapolitan, R. E., "*Probabilistic Reasoning in Expert Systems: Theory and Algorithms*," A Wiley-Interscience Publication, John Wiley & Sons, Inc., 1989

41. Neil, M; Fenton, N.; Nielsen, L., "Building Large-Scale Bayesian Networks," *The Knowledge Engineering Review*, Vol.15, No. 3, pp. 257-284, 2000

42. Norton, M. J., "Knowledge Discovery in Database," *Library Trends*, Vol. 48, No. 1, pp. 9-21, Summer 1999

43. Olesen, K. G., "Causal Probabilistic Networks with Both Discrete and Continuous Variables," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No. 3, pp. 275-279, March 1993

44. Pazzani, M. J., "Knowledge Discovery from Data," *IEEE Intelligent Systems & Their Applications*, Vol. 15, No. 2, pp. 10-13, 2000

45. Sarkar, S.; Murthy, I., "Constructing Efficient Belief Network Structures with Expert Provided Information," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 8, No. 1, pp. 134-143, Feb. 1996

46. Schulze-Kremer, S., "Discovery in the Human Genome Project," *Communication of the ACM*, Vol. 42, No. 11, pp. 62-64, 1999

47. Silberschatz, A.; Korth, H. F.; Sudarshan, S., *Database System Concepts*, 3rd edition, McGraw-Hill, New York, 1996

48. Speigelhalter, D. J.; Lauritzen, S. L., "Sequential Updating of Conditional Probabilities on Directed Graphical Structures," *Networks*, Vol. 20, pp. 579-605, 1990

49. Valdes-Perez, R. E., "Discovery Tools for Science Apps," *Communication of the ACM*, Vol. 42, No. 11, pp. 37-41, 1999

50. Ziarko, W., "Discovery through Rough Set Theory," *Communication of the ACM*, Vol.42, No.11, pp. 55-57, 1999