

A POLYNOMIAL TRANSFORMATION FROM VERTEX COVER PROBLEM TO  
EXACT INFERENCE PROBLEM IN BAYESIAN BELIEF NETWORKS

by  
MUSTAFA TACETTİN

Submitted to the Graduate School of Engineering and Natural Sciences  
in partial fulfillment of  
the requirements for the degree of  
Master of Science

Sabancı University  
Spring 2002

A POLYNOMIAL TRANSFORMATION FROM VERTEX COVER PROBLEM TO  
EXACT INFERENCE PROBLEM IN BAYESIAN BELIEF NETWORKS

APPROVED BY:

Dr. Tonguç Ünlüyurt .....  
(Dissertation Supervisor)

Assoc. Prof. Dr. Taner Bilgiç .....

Dr. Hüsnü Yenigün .....

DATE OF APPROVAL: .....

© Mustafa Tacettin 2002

All Rights Reserved

## **ABSTRACT**

Exact Inference problem in Belief Networks has been well studied in the literature and has various application areas. In this thesis, a polynomial time transformation from Vertex Cover Problem to Exact Inference problem in Belief Networks is proposed and proved. To understand and see the development of the transformation, some well-known transformations about Vertex Cover Problem and Exact Inference, are introduced. By using the transformation proposed, some Vertex Cover problems are converted to Exact Inference Problems and solved by softwares using the algorithms of Exact Inference.

## ÖZET

İnanç Ağında Olasılık Çıkarımı problemi literatürde üzerinde sıkça çalışılan ve uygulama alanı oldukça geniş olan bir problemdir. Bu tezde, Köşe Kapatma probleminden İnanç Ağında Olasılık Çıkarımı problemine polinom zamanda dönüşüm için bir yöntem önerildi ve bu yöntemin doğruluğu ispatlandı. Önerilen dönüşümü daha iyi anlamak için, Köşe Kapatma ve Olasılık Çıkarımı problemleriyle alakalı dönüşümler hakkında bilgi verildi. Önerilen dönüşüm ile ilgili olarak, bazı Köşe Kapatma problemleri Olasılık Çıkarımı problemine dönüştürüldü ve Olasılık Çıkarımı problemini çözen algoritmaları kullanan yazılımlarla çözüldü.

## **ACKNOWLEDGEMENTS**

I would like to thank Dr. Tongu Ünlüyurt, whom I learned lots of things. I would also like to thank Mr. Bilgic for the never-ending challenge during my work on the thesis, Caner for sharing his programming skills with me and Hacı Murat for his motivation.

## LIST OF FIGURES

Figure 2.1 A belief network structure corresponding to the example 3-SAT problem.....	12
Figure 2.2 An instance of VC.....	15
Figure 2.3 A belief network structure corresponding to the example VC problem.....	16
Figure 2.4 Vertex Nodes of BN .....	19
Figure 2.5 The parents of node $E_j$ where $E_j$ is $(V_i, V_z)$ .....	19
Figure 2.6 The Edge Result Nodes.....	20
Figure 2.7 The incoming arcs of the state node $S_{ij}$ .....	20
Figure 2.8 The State Result Nodes.....	21
Figure 2.9 Final Node denoted by Y .....	21
Figure 2.10 A VC Problem: Is it possible to have a vertex cover V where $ V  \leq 2$ ? .....	23
Figure 2.11 The BN representation of the example problem.....	24
Figure 2.12 The BN after Elimination of Barren Nodes.....	27
Figure 2.13 BN after conversion.....	29
Figure 2.14 Final BN for Node Elimination Method.....	29
Figure 2.15 Graphical Modification of Junction Tree Algorithm.....	31

## LIST OF TABLES

Table 2.1 Probability table for vertex nodes .....	22
Table 2.2 Probability table for edge nodes.....	22
Table 2.3 Probability table for edge result nodes.....	22
Table 2.4 Probability table for state nodes.....	23
Table 2.5 Probability table for state result nodes.....	23
Table 2.6 New Probability Table After Node Elimination .....	29
Table 2.7 Final Probability Table after Node Elimination Method .....	30
Table 3.1 Results of Experiment.....	38
Table 3.2 Results for $C/NON = 1$ .....	39



## TABLE OF CONTENTS

<b>1. INTRODUCTION .....</b>	<b>1</b>
<b>1.1 BASIC DEFINITIONS .....</b>	<b>2</b>
<b>1.2 LITERATURE REVIEW .....</b>	<b>6</b>
<b>2. TRANSFORMATIONS .....</b>	<b>10</b>
<b>2.1 TRANSFORMATION FROM 3-SAT .....</b>	<b>10</b>
<i>2.1.1 Transformation from 3-SAT to Exact Inference .....</i>	<i>10</i>
<i>2.1.2 Transformation from 3-SAT to VC .....</i>	<i>13</i>
<b>2.2 TRANSFORMATION FROM VC .....</b>	<b>16</b>
<i>2.2.1 Transformation from VC to Exact Inference for a special type of VC .....</i>	<i>16</i>
<i>2.2.2 Transformation from VC To Exact Inference .....</i>	<i>18</i>
2.2.2.1 Nodes of Belief Network .....	18
2.2.2.2 Edges of Belief Network .....	19
2.2.2.3 The Probability Tables .....	21
2.2.2.4 An Example Transformation .....	23
2.2.2.5 Correctness Of The Transformation .....	24
<b>2.3 SOLVING THE EXACT INFERENCE OBTAINED .....</b>	<b>26</b>
<i>2.3.1 Node Elimination Method .....</i>	<i>27</i>
<b>3.BENEFITS OF THE TRANSFORMATION .....</b>	<b>32</b>
<b>3.1 APPROXIMATION ALGORITHMS OF VC .....</b>	<b>32</b>
<b>3.2 USING SOFTWARES DEVELOPED FOR INFERENCE PROBLEM AS A VC SOLVER .....</b>	<b>35</b>
3.2.1 <i>Softwares Used .....</i>	35
3.2.2 <i>Random VC Creation .....</i>	36
3.2.3 <i>Converting VC to BN .....</i>	37
3.2.4 <i>Solving the Problem .....</i>	37
3.2.5 <i>Results .....</i>	37
<b>3.3 ANALYSIS OF THE RESULT .....</b>	<b>39</b>
<b>4. CONCLUSION &amp; FUTURE WORK .....</b>	<b>41</b>
<b>5.REFERENCES.....</b>	<b>43</b>

A POLYNOMIAL TRANSFORMATION FROM VERTEX COVER PROBLEM TO  
EXACT INFERENCE PROBLEM IN BAYESIAN BELIEF NETWORKS

by  
MUSTAFA TACETTİN

Submitted to the Graduate School of Engineering and Natural Sciences  
in partial fulfillment of  
the requirements for the degree of  
Master of Science

Sabancı University  
Spring 2002

A POLYNOMIAL TRANSFORMATION FROM VERTEX COVER PROBLEM TO  
EXACT INFERENCE PROBLEM IN BAYESIAN BELIEF NETWORKS

APPROVED BY:

Dr. Tonguç Ünlüyurt .....  
(Dissertation Supervisor)

Assoc. Prof. Dr. Taner Bilgiç .....

Dr. Hüsnü Yenigün .....

DATE OF APPROVAL: .....

© Mustafa Tacettin 2002

All Rights Reserved

## **ABSTRACT**

Exact Inference problem in Belief Networks has been well studied in the literature and has various application areas. In this thesis, a polynomial time transformation from Vertex Cover Problem to Exact Inference problem in Belief Networks is proposed and proved. To understand and see the development of the transformation, some well-known transformations about Vertex Cover Problem and Exact Inference, are introduced. By using the transformation proposed, some Vertex Cover problems are converted to Exact Inference Problems and solved by softwares using the algorithms of Exact Inference.

## ÖZET

İnanç Ağında Olasılık Çıkarımı problemi literatürde üzerinde sıkça çalışılan ve uygulama alanı oldukça geniş olan bir problemdir. Bu tezde, Köşe Kapatma probleminden İnanç Ağında Olasılık Çıkarımı problemine polinom zamanda dönüşüm için bir yöntem önerildi ve bu yöntemin doğruluğu ispatlandı. Önerilen dönüşümü daha iyi anlamak için, Köşe Kapatma ve Olasılık Çıkarımı problemleriyle alakalı dönüşümler hakkında bilgi verildi. Önerilen dönüşüm ile ilgili olarak, bazı Köşe Kapatma problemleri Olasılık Çıkarımı problemine dönüştürüldü ve Olasılık Çıkarımı problemini çözen algoritmaları kullanan yazılımlarla çözüldü.

## **ACKNOWLEDGEMENTS**

I would like to thank Dr. Tongu Ünlüyurt, whom I learned lots of things. I would also like to thank Mr. Bilgic for the never-ending challenge during my work on the thesis, Caner for sharing his programming skills with me and Hacı Murat for his motivation.

## LIST OF FIGURES

Figure 2.1 A belief network structure corresponding to the example 3-SAT problem.....	12
Figure 2.2 An instance of VC.....	15
Figure 2.3 A belief network structure corresponding to the example VC problem.....	16
Figure 2.4 Vertex Nodes of BN .....	19
Figure 2.5 The parents of node $E_j$ where $E_j$ is $(V_i, V_z)$ .....	19
Figure 2.6 The Edge Result Nodes.....	20
Figure 2.7 The incoming arcs of the state node $S_{ij}$ .....	20
Figure 2.8 The State Result Nodes.....	21
Figure 2.9 Final Node denoted by Y .....	21
Figure 2.10 A VC Problem: Is it possible to have a vertex cover V where $ V  \leq 2$ ? .....	23
Figure 2.11 The BN representation of the example problem.....	24
Figure 2.12 The BN after Elimination of Barren Nodes.....	27
Figure 2.13 BN after conversion.....	29
Figure 2.14 Final BN for Node Elimination Method.....	29
Figure 2.15 Graphical Modification of Junction Tree Algorithm.....	31



## LIST OF TABLES

Table 2.1 Probability table for vertex nodes .....	22
Table 2.2 Probability table for edge nodes.....	22
Table 2.3 Probability table for edge result nodes.....	22
Table 2.4 Probability table for state nodes.....	23
Table 2.5 Probability table for state result nodes.....	23
Table 2.6 New Probability Table After Node Elimination .....	29
Table 2.7 Final Probability Table after Node Elimination Method .....	30
Table 3.1 Results of Experiment.....	38
Table 3.2 Results for $C/NON = 1$ .....	39

## TABLE OF CONTENTS

<b>1. INTRODUCTION .....</b>	<b>1</b>
<b>1.1 BASIC DEFINITIONS .....</b>	<b>2</b>
<b>1.2 LITERATURE REVIEW .....</b>	<b>6</b>
<b>2. TRANSFORMATIONS .....</b>	<b>10</b>
<b>2.1 TRANSFORMATION FROM 3-SAT .....</b>	<b>10</b>
<i>2.1.1 Transformation from 3-SAT to Exact Inference .....</i>	<i>10</i>
<i>2.1.2 Transformation from 3-SAT to VC .....</i>	<i>13</i>
<b>2.2 TRANSFORMATION FROM VC .....</b>	<b>16</b>
<i>2.2.1 Transformation from VC to Exact Inference for a special type of VC .....</i>	<i>16</i>
<i>2.2.2 Transformation from VC To Exact Inference .....</i>	<i>18</i>
2.2.2.1 Nodes of Belief Network .....	18
2.2.2.2 Edges of Belief Network .....	19
2.2.2.3 The Probability Tables .....	21
2.2.2.4 An Example Transformation .....	23
2.2.2.5 Correctness Of The Transformation .....	24
<b>2.3 SOLVING THE EXACT INFERENCE OBTAINED .....</b>	<b>26</b>
<i>2.3.1 Node Elimination Method .....</i>	<i>27</i>
<b>3.BENEFITS OF THE TRANSFORMATION .....</b>	<b>32</b>
<b>3.1 APPROXIMATION ALGORITHMS OF VC .....</b>	<b>32</b>
<b>3.2 USING SOFTWARES DEVELOPED FOR INFERENCE PROBLEM AS A VC SOLVER .....</b>	<b>35</b>
3.2.1 <i>Softwares Used .....</i>	35
3.2.2 <i>Random VC Creation .....</i>	36
3.2.3 <i>Converting VC to BN .....</i>	37
3.2.4 <i>Solving the Problem .....</i>	37
3.2.5 <i>Results .....</i>	37
<b>3.3 ANALYSIS OF THE RESULT .....</b>	<b>39</b>
<b>4. CONCLUSION &amp; FUTURE WORK .....</b>	<b>41</b>
<b>5.REFERENCES.....</b>	<b>43</b>

## 1. INTRODUCTION

We are living in the world of causes and effects. An event is a result of action and all actions are the supplementary results of decisions. While giving ordinary decisions, maybe unconsciously we are checking the outer causes and effects about our decision. Science gives us the opportunity to model the cause and effect relations. Belief networks constitute such a framework that enables us to demonstrate the causal relationships between the events. In real life, the relations between the events are not deterministic, so in a belief network the effects of causes are determined by probabilities.

With such a construction, it is possible to find out the probability of an event when a certain indirectly related other event happens. Calculating the probability of an event given related evidences is called Exact Inference Problem. The methods for solving these type of problems are important for troubleshooting, medical diagnosis, etc.

Exact Inference Problem is very difficult to solve. But, in the last decade lots of scientists have worked on this subject. Another interesting problem in the literature is Vertex Cover problem (VC). The details of the problem can be found in the following sections. Our target in this thesis is to show that VC is an instance of Exact Inference Problem in BNs. Finding a polynomial time transformation from VC to Exact Inference is enough to prove that, these two problems are in the same class of complexity and VC is an instance of Exact Inference Problem.

Polynomial time transformations are so important that it enables to view a problem

in a different manner. Many people work on these transformations. They attempt to find the analogies among the problems in order to use the algorithms developed in a research area to another field. Another aim of transformation among the well known problems can be using the heuristics developed independently for each problem for the other type of problem.

By converting the problem structure, it is possible to use the heuristics for Exact Inference Problem to VC. After the transformation, the structure of the Belief Network for the Exact Inference problem is investigated to obtain better algorithms or some shortcuts for the known algorithms in the literature. Also, the approximations used for VC can be applied to certain Exact Inference Problem instances.

In the following sections, after making the definitions of the terms used and the literature review, some known transformations about Exact Inference and VC will be introduced. Then, in the succeeding section, the proposed method and its usage will be described. Finally, application of the transformation proposed and its results will be explained.

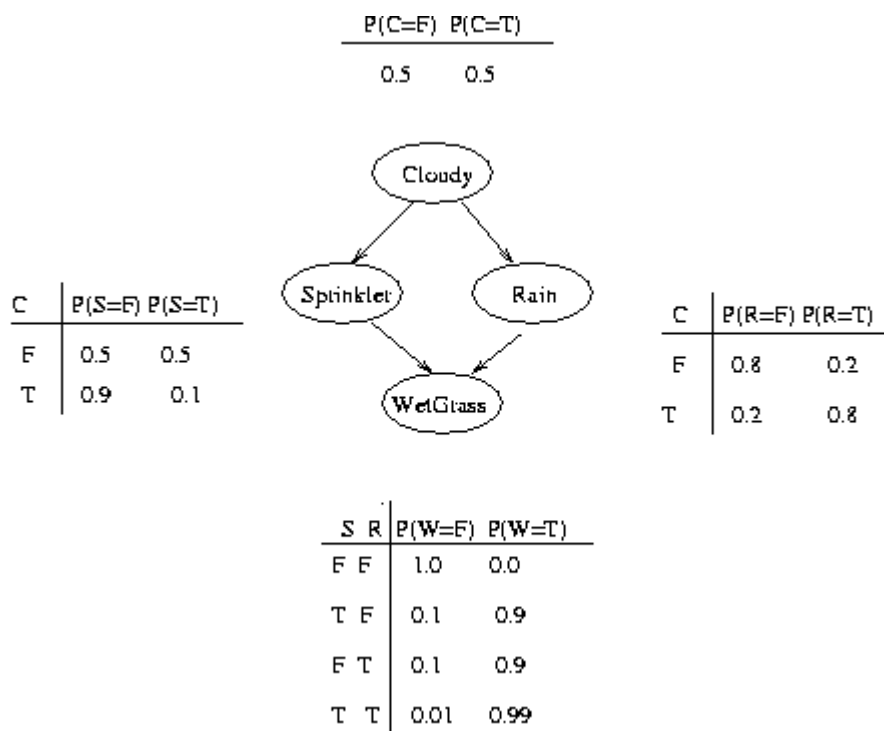
## 1.1 Basic Definitions

**Definition 1.** A “Belief Network” is a graph that consists of a set of random variables, a set of directed links connecting pairs of nodes where each node has a conditional probability table that quantifies the effects that the parents have on the node. The set of random variables for the Belief Network is the union of the states of the nodes. A state of the node is the random variable that demonstrates the condition or the value of the node. The graph has no directed cycles so it is a directed acyclic graph (DAG). The causal relationships between particular variables are represented on the graph as nodes. Nodes are connected by causal links represented by arrows, that points from parent nodes (causes) to child nodes (effects).

**Definition 2.** “Exact Inference Problem” on a belief network is the process of

computing  $Pr( V=v / E=e )$ , or simply  $Pr(v / e)$  where  $v$  is a value of a variable  $V$  and  $e$  is an assignment of values to a set of variables  $E$  in the belief network.  $e$  is also called evidence or observation.

To understand the Exact Inference problem, a simple example will be introduced.



In this example all nodes are binary, i.e., all nodes have two possible values, that will be denoted by T (*true*) and F (*false*).

We see that the event "grass is wet" ( $W=true$ ) has two possible causes: either the water sprinkler is on ( $S=true$ ) or it is raining ( $R=true$ ). The strength of this relationship is shown in the table. For example, we see that  $Pr(W=true / S=true, R=false) = 0.9$  (second row), and hence,  $Pr(W=false / S=true, R=false) = 1 - 0.9 = 0.1$ , since each row must sum to one. Since the node C has no parents, its probability table specifies the prior probability that it is cloudy (in this case, 0.5). An instance of the Exact Inference problem can be computing the probability of sprinkler is on where grass is wet and it is raining.

$$Pr(S=true/W=true,R=true)$$

How to find the answer is the art of Artificial Intelligence and it says that the answer is 0.1945.

**Definition 3.** By using the notation of (Garey, Johnson, 1979), a “polynomial transformation” from a language  $L1$  to a language  $L2$  is a function  $f$  that satisfies the following two conditions:

- 1- There is a polynomial time program that computes  $f$ .
- 2-  $x \in L1$  if and only if  $f(x) \in L2$ .

If there is a polynomial transformation from  $L1$  to  $L2$ , it is written  $L1 \propto L2$  and read as “ $L1$  transforms to  $L2$ ”.

One can make the following observations about polynomial transformations:

- 1- If  $L1 \propto L2$  and  $L2 \in P$ , then  $L1 \in P$ .
- 2- If  $L1 \notin P$  then  $L2 \notin P$ .
- 3- if  $L1 \propto L2$  and  $L2 \propto L3$ , then  $L1 \propto L3$ .

The advantages of these transformations are; if a problem is converted into another form, all the heuristics developed individually in for the target problem can be applied to the converted problem. In our case, if it is possible to find a polynomial transformation from Vertex Cover problem to the Exact Inference in BN problem, any algorithm developed or any software developed for Exact Inference problem in BN can be applied to VC.

**Definition 4.** The problems are said to be “polynomially equivalent” whenever both  $L1 \propto L2$  and  $L2 \propto L1$  holds. That means if one problem has a polynomial time algorithm so does the other.

**Definition 5.** “Easy Problem”s are the problems which can be solved by such an algorithm whose complexity is polynomial, that is  $O(f)$  for a polynomial  $f$  in the size of input data.

**Definition 6.** The problems whose solution is either ‘yes’ or ‘no’ is said to be a “decision problem”, or “logic problem”.

Hamiltonian Cycle (HC) problem is an example for such a decision problem. Or, the question whether a directed graph is acyclic is a decision problem. We know that, HC is NP-Complete.

**Definition 7.** The problems where an optimal solution is searched with respect to a certain criterion, is “optimization problem”.

Traveling Salesman Problem (TSP) is an example to the optimization problem. (Jungnickel,1999) says that each optimization problem corresponds to a decision problem. This can be illustrated using TSP as follows: For a given matrix  $W = (w_{ij})$  and a positive integer  $M$ , the corresponding decision problem is the question whether there exists a tour  $\pi$  such that  $w(\pi) \leq M$  or not. Another type of problem between optimization problems and decision problems is said to be evaluation problems, where the value of an optimization problem is asked for, without requiring the explicit solution itself. Actually, any optimization problem can be converted to an evaluation problem. Any algorithm that solves the optimization problem also solves the evaluation problem. Analogously, solving an evaluation problem also gives a solution for the associated decision problem.

It is easier to deal with the decision problems while converting it to a BN problem, because the answer required from the Exact Inference (EI) problem will be either 0 or 1. Namely, either the probability of a an event is higher than a threshold value or not. So, while converting the “Vertex Cover Problem” to EI, the decision problem version of the vertex cover problem is used.

**Definition 8.** A “vertex cover” of an undirected graph  $G = (V,E)$  is a subset  $S$  of  $V$  such that if  $(u, v)$  is an edge of  $G$ , then either  $u \in S$  or  $v \in S$  (or both). While talking about the VC problem in this content, the formal definition is: “Let  $G = (V,E)$  be a graph and  $k$  a positive integer. Does  $G$  have a vertex cover  $V'$  with  $|V'| \leq k$ ?”.

**Definition 9.** The definition for “3-SAT problem” is as follows: Consider a collection  $C=\{c_1,c_2,\dots,c_m\}$  of clauses of a finite set  $U$  of  $n$  boolean variables. Let (the literal)  $u$  be true if and only if the variable  $u$  is true and let  $\neg u$  be true iff  $u$  is false. Each clause  $c_i$  contains a disjunction of three literals over  $U$ . A collection  $C$  of clauses over  $U$  is satisfiable if and only if there exists some truth assignment for  $U$  that simultaneously satisfies all the clauses in  $C$ . The 3-SAT decision problem involves determining whether there is a truth assignment for  $U$  that satisfies all the clauses in  $C$ .

## 1.2 Literature Review

To show that a problem is in NP-Complete set, the method used in the literature is, showing a known NP-Complete problem can be polynomially transformed into that problem. The transformations among, Exact Inference problem and 3-SAT problem (Cooper,1990); VC (Vertex cover) problem and 3-SAT(Garey, Johnson,1979); HC and 3-SAT(Papadimitriou,1982); HC and VC (Garey, Johnson,1979); Clique problem and VC (Homer, Selman, 2001); HC and TSP are proven. One other transformation is MMP (Multidepot Multisalesman Problem) to TSP. Any algorithm of TSP can be used to solve the transformed TSP (Guoxing,1995). But there is no such transformation in literature directly (without the detour using 3-SAT) from Exact Inference to VC.

The techniques used for proving NP-completeness results vary almost as widely as the NP-complete problems themselves. However, there are several general types of proofs that occur frequently and that can provide a suggestive framework for deciding how to go about proving a new problem is NP-complete. These are restriction, local replacement and component design (Garey, Johnson, 1979). An NP-completeness proof by restriction for a given problem  $A \in NP$  consists simply of showing that  $A$  contains a known NP-complete problem  $B$  as a special case. In proofs by local replacement, all we do is pick some aspect of the known NP-complete problem instance to make up a collection of basic units, and we obtain the corresponding instance of the target problem by replacing each basic unit, in a uniform way, with a different structure. The basic idea for component design is to use the



constituents of the target problem instance to design certain components that can be combined to realize instances of the known NP-complete problem. The transformation from VC to Exact Inference that we will introduce is in the form of component design, which is the most complicated (Garey, Johnson 1979).

As is becoming increasingly well known, an influence diagram was defined in the 1970s as a graphical representation of the relationship of the decisions and uncertainties in a decision problem (Howard,1990). An influence diagram is a graphical structure used to model uncertain variables and decisions and to explicitly reveal probabilistic dependence and the flow of data. It is an intuitive framework to formulate problems as seen by decision makers and to incorporate the knowledge of experts. It also is a precise description of information that can be stored and manipulated by a computer. An algorithm is developed that can evaluate any well-formed influence diagram and determine the optimal policy for its decisions. Since the diagram can be examined directly, there is no need to construct other representations, such as a decision tree. As a result, the examination can be performed using the decision maker's perspective on the problem. Questions concerning sensitivity and the value of information are natural and easily posed. Modifications to the model indicated by such analyses can be made directly to the problem formulation and then evaluated directly (Shachter, Ross, 1988).

It has since become the most effective tool available for the representation and evaluation of decision problems. Researchers are extending the capability of the tool while practitioners are expanding its use in aiding decision-makers (Howard, 1990). The field of Belief Networks, and graphical models in general, has grown enormously over the last fifteen years, with theoretical and computational developments in many areas. As a consequence there is now a fairly large set of theoretical concepts and results for newcomers to the field to learn (Cowell, 1999).

(Cooper, 1990) states that Belief Networks provide a natural, efficient method for representing probabilistic dependencies among a set of variables. For these reasons, numerous researchers are exploring the use of belief networks as a knowledge

representation in artificial intelligence. Algorithms have been developed for efficient probabilistic inference using special classes of belief networks. More general classes of belief networks, however, have eluded efforts to develop efficient inference algorithms. Probabilistic inference problem using Belief Networks is NP-hard. Therefore, it seems unlikely that an exact algorithm can be developed to perform probabilistic inference efficiently over all classes of belief networks.

(Dagum, Luby, 1993) showed that approximating probabilistic inference in Belief Networks is NP-hard. They again used the reduction of Cooper, and show that for all  $\epsilon < 1/2$ , there is no polynomial-time absolute approximation algorithm for  $Pr[V = v / E = e]$  if  $NP \neq P$ . The absolute approximation gives an estimate of  $Z$  where

$$Pr[V = v / E = e] - \epsilon \leq Z \leq Pr[V = v / E = e] + \epsilon$$

So  $\epsilon$  is the maximum error of the absolute approximation.

Also, if  $NP \neq P$ , there is no polynomial-time randomized absolute approximation algorithm for  $Pr[V = v / Y = y]$  where  $\epsilon < 1/2$  and  $\delta < 1/2$  where  $\delta$  is the probability of failure for the randomized approximation algorithm in the specified bounds of  $Z$  determined by  $\epsilon$ . (Dagum, Luby, 1997) showed that there is a polynomial time algorithm while approximating probabilistic inference, if the Belief Network contains probabilities that come arbitrarily close to zero in the case that the evidence set is empty or constant-sized and the conditions for the error bounds are the same as above.

Another issue about Exact Inference problem is to group Belief Networks according to the groups complexity. It is shown that even for suprisingly restricted cases, the problem is NP-hard (Roth, 1996). It is also NP-hard when you convert it to Approximate Inference Problem. The place of Exact Inference Problem among the NP-hard problems is also worked by Roth in order to rank the problem among the others.

The related problems with Belief Networks such as MAP explanation was shown to be NP-hard for exact solution (Shimony,1994) and for approximation (Abdelbar,1997). MAP explanation is very similar to Exact Inference problem. For MAP explanation, the

objective is to find the instantiation  $I$  with probability  $Pr(A / e)$ . The MAP explanation with bounded probabilities is also NP-hard. This is shown by (Abdelbar,2000).

Belief Networks have many different application areas. They provide a powerful tool for simulating the interactions between physical, social and economic variables. Although belief networks are no substitute for high quality fieldwork, it is clear that they provide a mathematical framework that facilitates interdisciplinary data capture and analysis (Batchelor,1999). It has applications nearly in all diciplines such as finance, troubleshooting, medical diagnosis and agriculture.

## 2. TRANSFORMATIONS

### 2.1 Transformation from 3-SAT

#### 2.1.1 Transformation from 3-SAT to Exact Inference

By following Cooper(1990) and Bilgic(2002), it is possible to convert a 3-SAT problem to decision problem version of the Exact Inference problem.

Let's consider the following instance of 3-SAT with  $U=\{u_1, u_2, u_3, u_4\}$  and

$$C=\{(u_1 \vee u_2 \vee u_3), (-u_1 \vee -u_2 \vee u_3), (u_2 \vee -u_3 \vee u_4)\}$$

The truth assignment  $u_1 = T$ ,  $u_2 = F$ ,  $u_3 = F$ , and  $u_4 = T$  results in the answer “yes” for this example.

For the probabilistic inference problem, let's assume without loss of generality that all the variables can take only two values,  $D_i = \{T, F\}$  for all  $i$  where  $D_i$  is the set of states of node  $i$ . Furthermore, considering the inference problem without the introduction of new evidence  $e$ , i.e., we are interested in the marginal probability  $Pr(Y)$  rather than  $Pr(Y/e)$ . If it is possible to show that a restricted version of the problem is NP-hard, the more general version of the problem will also be NP-hard.

The decision problem version of the inference problem will return “yes” if

$$Pr(Y = T) > 0,$$

and “no” otherwise. Let’s denote this decision problem as Inference in Belief Networks Decision Problem (IBND). We will transform 3-SAT to IBND.

Let  $U = \{u_1, u_2, \dots, u_n\}$  and  $C = \{c_1, c_2, \dots, c_m\}$  be any instance of 3-SAT. The BN constructed with a variable  $Y$  such that  $Pr(Y=T) > 0$  means  $C$  is satisfiable.

The BN corresponding to the 3-SAT will have several components:

1. A truth setting component
2. A clause satisfaction testing component
3. An overall satisfaction testing component

Figure 2.1 depicts the BN corresponding to the example 3-SAT problem.

The BN is represented as  $(G, P)$  where  $G=(N, A)$  is the DAG composed of nodes(vertices)  $N$  and arcs (edges)  $A$ . The truth setting component of the BN corresponding to 3-SAT is given as  $((N_t, \emptyset), P_t)$  and it is defined for all variables in  $U$ . In particular  $N_t$  is the set of all variables from  $U$  and  $P_t$  is the set of probabilities set at  $1/2$ :

$$N_t = U,$$

and,

$$P_t = \{Pr(u_1 = T) = 1/2, Pr(u_2 = T) = 1/2, \dots, Pr(u_n = T) = 1/2\}.$$

See Figure 2.1 where four nodes are generated as shown in the first row of the network.

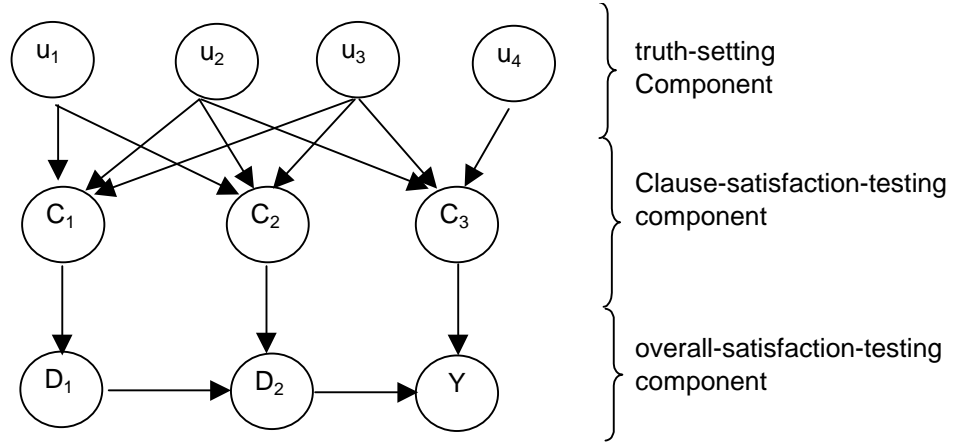


Figure 2.1 A belief network structure corresponding to the example 3-SAT problem

For each clause  $c_j \in C$  of the 3-SAT problem ( $1 \leq j \leq m$ ), there is a clause satisfaction testing sub component  $((N_j^s, A_j^s), P_j^s)$  that tests whether a given instantiation of the variables in  $U$  satisfies the clause  $c_j \in C$ . The components of  $((N_j^s, A_j^s), P_j^s)$  are defined as follows:

$$N_j^s = \{w_j^1, w_j^2, w_j^3, C_j\},$$

Where  $w_j^1$  ( $w_j^2, w_j^3$ ) is the variable corresponding to the first (second, third) literal in clause  $c_j$ . From the example  $c_2 = (-u_1 \vee -u_2 \vee u_3)$  and therefore  $w_2^1 = u_1$ ,  $w_2^2 = u_2$ , and  $w_2^3 = u_3$ . The variable  $C_j$  represents the truth value of clause  $c_j$ .

$$A_j^s = \{(w_j^1, C_j), (w_j^2, C_j), w_j^3, C_j\},$$

$$P_j^s = \{Pr(C_j = T | \pi_{c_j})\}$$

where  $\pi_{c_j}$  represents the conjunction of the three variables  $w_j^1, w_j^2, w_j^3$  of clause  $c_j$  and

$$Pr(C_j = T | \pi_{c_j}) = \begin{cases} 1 & \text{if } g_j(\pi_{c_j}) = T, \\ 0 & \text{if } g_j(\pi_{c_j}) = F, \end{cases}$$

where  $g_j(\pi_{c_j})$  is the truth function for clause  $c_j$ .

From the example problem, for the clause  $c_3 = (u_2 \vee -u_3 \vee u_4)$ . There is a sub-component

$((N_3^s, A_3^s), P_3^s)$ , where

$$N_3^s = \{u_2, u_3, u_4, C_3\}$$

$$A_3^s = \{(u_2, C_3), (u_3, C_3), (u_4, C_3)\}$$

$$P_3^s = \{Pr(C_3 = T | u_2 = T, u_3 = T, u_4 = T) = 1\},$$

$$\begin{aligned}
&Pr(C_3 = T | u_2 = T, u_3 = T, u_4 = F) = 1, \\
&Pr(C_3 = T | u_2 = T, u_3 = F, u_4 = T) = 1, \\
&Pr(C_3 = T | u_2 = T, u_3 = F, u_4 = F) = 1, \\
&Pr(C_3 = T | u_2 = F, u_3 = T, u_4 = T) = 1, \\
&Pr(C_3 = T | u_2 = F, u_3 = T, u_4 = F) = 0, \\
&Pr(C_3 = T | u_2 = F, u_3 = F, u_4 = T) = 1, \\
&Pr(C_3 = T | u_2 = F, u_3 = F, u_4 = F) = 1 \}
\end{aligned}$$

The clause satisfaction testing component is composed of the union of such sub components:

$$N^s = \cup_{j=1}^m N_j^s, \quad A^s = \cup_{j=1}^m A_j^s, \quad P^s = \cup_{j=1}^m N_j^s.$$

Finally, the overall satisfaction testing component  $((N^0, A^0), P^0)$  tests whether all of  $m$  clauses are satisfied simultaneously. In particular, there is an arc from each  $C_j$  to a variable  $Y$  and the probability  $Pr(Y=T | C_1, C_2, \dots, C_m)$  is 1 if and only if  $C_1 = C_2 = \dots, C_m = T$ , otherwise it is 0.

For the BN constructed in Figure 2.1, it is achieved via intermediate dummy nodes,  $D_i$ . Each dummy variable  $D_i$  has the value  $T$  if each of its parents has the value  $T$ ; otherwise it has the value  $T$  with probability 0.

This construction of BN can be performed in polynomial time. The construction of the truth setting component is  $O(n)$ , the clause satisfaction testing component is  $O(m)$ , and the overall satisfaction component is  $O(m)$ . The result for this transformation is the clause set  $C$  is satisfiable if and only if  $Pr(Y = T) > 0$ .

### 2.1.2 Transformation from 3-SAT to VC

By following (Jungnickel, 1999), it is possible to transform a 3-SAT problem to Vertex Cover problem.

Let  $C_1, \dots, C_m$  be an instance of 3-SAT,  $x_1, \dots, x_n$  are the variables occurring in  $C_1, \dots, C_m$ . For each variable  $x_i$ , consider a copy of the complete graph  $K_2$ , namely

$$T_i = (V_i, E_i) \text{ where } V_i = \{x_i, x_i'\} \text{ and } E_i = \{x_i x_i'\};$$

The purpose of these sub-graphs is to determine the Boolean value of  $x_i$ . Analogously, for each clause  $C_j$  ( $j = 1, \dots, m$ ), we define a copy  $S_j = (V_j', E_j')$  of  $K_3$ , where

$$V_j' = \{c_{1j}, c_{2j}, c_{3j}\} \text{ and } E_j' = \{c_{1j}c_{2j}, c_{1j}c_{3j}, c_{2j}c_{3j}\};$$

The purpose of these satisfaction-testing components is to check the Boolean value of the clauses. Note that each vertex cover of  $G$  has to contain, for each  $j$ , at least two of the three vertices in  $V_j'$ .

The graphs  $T_i$  ( $i = 1, \dots, n$ ) and  $S_j$  ( $j = 1, \dots, m$ ) are the 'special components' of our graph  $G$ ; they do not depend on the explicit structure of the term  $C_1, \dots, C_m$ , but only on  $n$  and  $m$ . The only part of the construction of  $G$  where the literals occurring in the clauses are used is the part we turn to now: Fixing the edges connecting the  $S_j$  and the  $T_i$  ('communication edges'). For each clause  $C_j$ , let  $u_j, v_j$  and  $w_j$  be three literals occurring in  $C_j$ , and define a set of edges

$$E_j'' = \{c_{1j}u_j, c_{2j}v_j, c_{3j}w_j\}.$$

Finally, we define  $G = (V, E)$  by

$$V := \bigcup_{i=1}^n V_i \cup \bigcup_{j=1}^m V_j' \quad \text{and} \quad E := \bigcup_{i=1}^n E_i \cup \bigcup_{j=1}^m E_j' \cup \bigcup_{j=1}^m E_j'',$$

and put  $k = n + 2m$ . Obviously, the construction of  $G$  can be performed in polynomial time (in  $n$  and  $m$ ). Figure 2.2 shows as an example, the graph corresponding to instance of 3-SAT.

$$(x_1 + x_3' + x_4') (x_1' + x_2 + x_4')$$



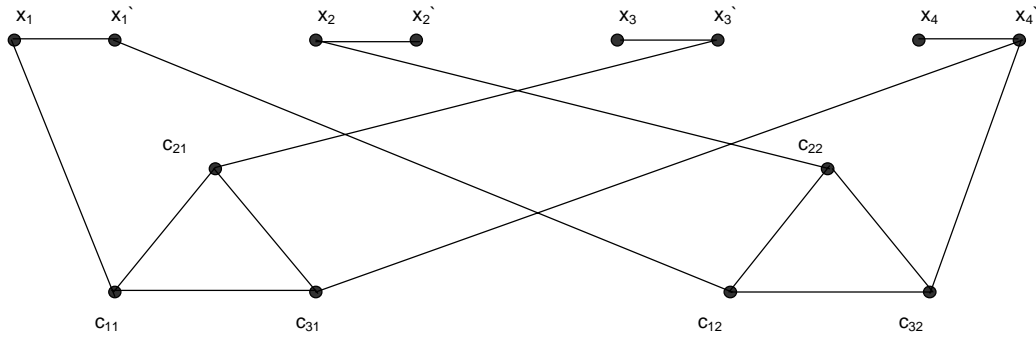


Figure 2.2 An instance of VC

The claim is,  $G$  has a vertex cover  $W$  with  $|W| \leq k$  if and only if there is a combination of values for  $x_1, \dots, x_n$  such that  $C_1, \dots, C_m$  has value true. Any vertex cover has to contain at least  $n + 2m = k$  vertices so that  $|W| = k$  is achieved. Moreover, we know that, if such  $W$  exists, it has to contain, for each  $i$ , exactly one of the vertices  $x_i$  and  $x_i'$  and for each  $j$ , exactly two of the three vertices of  $S_j$ .

Now, suppose  $W$  is such a vertex cover. Then we can use  $W$  as follows to obtain a combination  $w$  of Boolean values for the variables  $x_1, \dots, x_n$ : If  $W$  contains  $x_i$ , we set  $w(x_i) = \text{true}$ ; otherwise  $W$  has to contain the vertex  $x_i'$  and we set  $w(x_i) = \text{false}$ . Now consider an arbitrary clause  $C_j$ . As  $W$  contains exactly two of three vertices in  $V_j'$ , these two vertices are incident with exactly two of three edges in  $E_j''$ . As  $W$  is vertex cover, it has to contain a vertex incident with third edge ( $c_{3j}w_j$ , for the example above), and hence  $W$  contains corresponding vertex in one of the  $V_i$  (in our example, the vertex corresponding to the literal  $w_j$ , that is, either  $x_i$  or  $x_i'$ ). According to our definition of assignment  $w$  of Boolean values, this literal has the value true, so that the clause  $C_j$  also is true. As this holds for all  $j$ , the combination  $w$  of Boolean values gives the term  $C_1, \dots, C_m$  also the Boolean value true.

Conversely, let  $w$  be an assignment of Boolean values for the variables  $x_1, \dots, x_n$  such that  $C_1, \dots, C_m$  takes the value true. We define a subset  $W \subset V$  as follows: If  $w(x_i) = \text{true}$ ,  $W$  contains vertex  $x_i$ , otherwise  $W$  contains  $x_i'$  (for  $i = 1, \dots, n$ ). Then, all edges in  $E_i$  are covered. Moreover, for each clause  $C_j$  (which has value true using  $w$ ), at least one edge  $e_j$  of  $E_j''$  is covered. Adding the end vertices  $S_j$  of the other two edges of  $E_j''$  to  $W$ , obviously

all edges of  $E_j''$  and of  $E_j'$  are covered and  $W$  is a vertex cover of cardinality  $k$ .

## 2.2 Transformation from VC

### 2.2.1 Transformation from VC to Exact Inference for a special type of VC

By using these two transformations, it is possible to say that the problems in a special form for VC can be converted to BN. The special form is the graph obtained after the transformation from 3-SAT to VC.

Suppose the graph for the VC satisfies the following conditions. There are two type of nodes for the graph. These are clause nodes and decision nodes. Decision nodes are in pairs, there is an edge between these two nodes. The clause nodes are grouped in triples. For each group, each vertex in the group has an edge with the other two node, and there is an edge which combines the node with a decision node. For a single clause group that consist of 3 nodes, there are 3 decision node connected, and these 3 node are in different decision groups. For the graph there is no other node or edge.

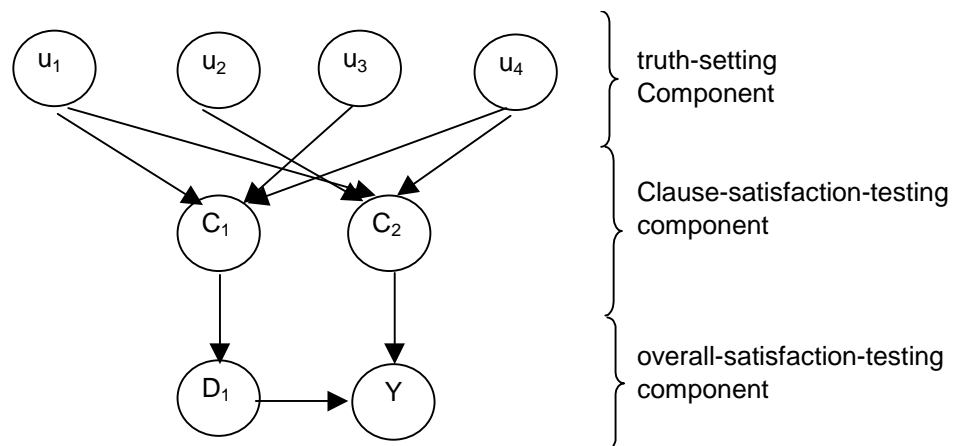


Figure 2.3 A belief network structure corresponding to the example VC problem

VC problem in such a graph can be converted to an inference problem by using same

technique described above. The graph is a representation of a 3-SAT problem. Finding the original 3-SAT problem will yield the Exact Inference problem. An example problem is as follows:

Lets take the graph in figure 2.2 as the vertex cover problem with cardinality  $k = 8$  which is  $n + 2m$ . This is actually the representation of the example 3-SAT problem where the clauses are  $(x_1 + x_3 + x_4)$   $(x_1 + x_2 + x_4)$

We can construct the Exact Inference problem with BN for this sample problem. Figure 2.3 shows the BN representation of the 3-SAT problem, so it represents the VC problem in Figure 2.2.

If we solve this BN for  $Pr(Y = T)$ , the probability will be greater than 0, means that for a particular assignment of nodes to correct values it is possible to achieve  $Y$  is true. Then, for the logic problem VC, we can say that there exists a solution with vertex cover of cardinality 8. Then, what is wrong with this transformation?

Firstly, to apply this transformation, the analysis of the original graph for VC should be done. By this analysis, whether the graph is suitable for the transformation condition is answered. This analysis has a cost and a formal way to do must be determined.

This transformation is not the required transformation. Actually, the transformation that is sought should be applied for all type of VC problems.

Besides, covering all aspects of VC problem, another important fact to be considered for the transformation is doing in a reasonable time. While performing it, the complexity should be polynomial.

## 2.2.2 Transformation from VC To Exact Inference

In this section, we will transform the general VC problem to Exact Inference problem in polynomial time. The transformation should include any instance of VC problem. While converting VC, the Belief Network created will be in a fix structure. The Belief Network contains nodes, edges and probability tables. These nodes and edges can be classified into groups. Let's consider a graph  $G=(V,E)$  for the vertex cover problem, with  $V=\{V_1,\dots,V_n\}$  and  $E=\{E_1,\dots,E_m\}$ .

### 2.2.2.1 Nodes of Belief Network

The nodes of the Belief Network are defined as follows:

**Vertex nodes:** For each vertex  $V_i$  on the graph, construct a vertex node for the belief network. So there are  $n$  vertex nodes. Figure 2.4 shows the vertex nodes.

**Edge nodes:** For each edge  $E_i$  on the graph, construct a node for the belief network. So there are  $m$  edge nodes. Figure 2.5 shows the edge nodes.

**Edge Result node:** For each edge on the graph, except the edge with max. index number, construct a node for the belief network to see whether the edge is wrapped for the original problem or not. So, there are  $m-1$  edge result node. Figure 2.6 shows the edge result nodes.

**State node:** Corresponding to the number of vertices in the vertex cover, construct state nodes for the belief network that counts the number of vertices in the vertex cover. For a vertex node  $V_i$  there are  $i+1$  state nodes. Figure 2.7 shows a state node  $S_{ij}$  where  $0 \leq j \leq i$ .

**State Result nodes:** To see whether the number of covered vertices is less than or equal to  $k$  or not, construct the state result nodes. There are  $k+1$  state result nodes. Figure 2.8 shows the state result nodes.

**Final node:** The node for BN that is associated with whether the solution for VC is true or false. Figure 2.9 shows the final node.

For each node, there are two states, which is true or false. Total number of nodes is  $(2m + k + [(n+2) * (n + 1) / 2 ])$

### 2.2.2.2 Edges of Belief Network

While describing the edges of the Belief Network, the relation of a group of node and its parents will be utilized.



**For Vertex nodes:** These nodes don't have parent nodes.

Figure 2.4 Vertex Nodes of BN

**For Edge Nodes:** Each edge node has two parent nodes. These are the vertex nodes that form the edge in the original graph.

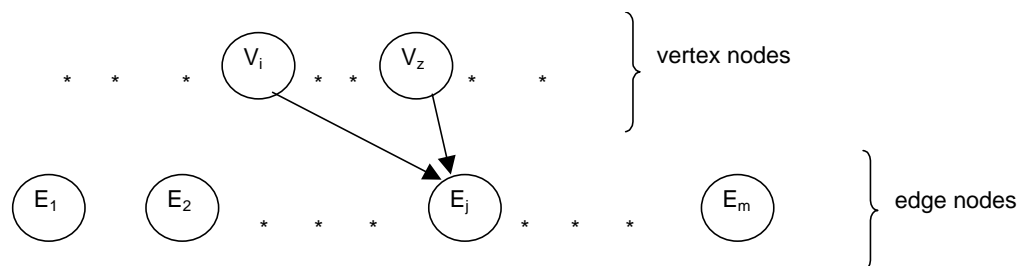


Figure 2.5 The parents of node  $E_j$  where  $E_j$  is  $(V_i, V_z)$

**For Edge Result nodes:** Each edge result node have at most two parent nodes. These are the predecessor edge result node and an edge node.

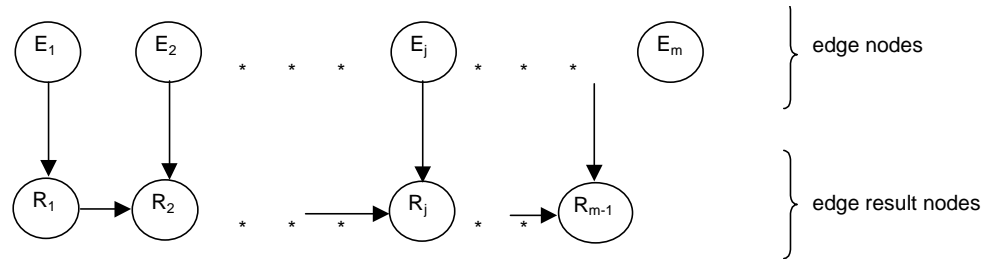


Figure 2.6 The Edge Result Nodes

**For State nodes:** The state nodes can be classified into  $n$  groups. Each group can be matched to a vertex node. In each group there are  $x$  nodes, where  $x$  is the order of the vertex node matched to that group. So, total number of state nodes is  $[(n+1)*(n+2)/2]-1$ . For each state node, there are at most 3 parent nodes.

The notation used for a state node is  $S_{ij}$ , where  $i$  is the vertex node illustrated with this node, and the  $j$  is the state of the first  $i$  vertex means if  $S_{ij}$  is true,  $j$  of the first  $i$  vertex is covered. It is obvious that,  $j \leq i$ .  $j$  can be 0.

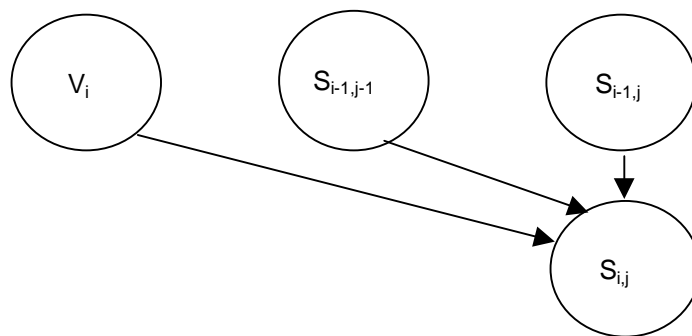


Figure 2.7 The incoming arcs of the state node  $S_{ij}$

The incoming arcs for state  $S_{ij}$  are added if applicable. In Figure 2.7, the arcs are possible to draw if  $S_{i-1,j}$  and  $S_{i-1,j-1}$  exists. If they don't exist, then the arcs are not in BN.

However, at least there is always one arc to add, which comes from  $V_i$ .

**For State Result nodes:** The state result nodes are constructed to see the total number of covered vertex. There are  $k+1$  of them. For each result node there are at most 2 parent nodes. These are the predecessor node and the state node  $S_{nj}$ , where  $j \leq k$ .

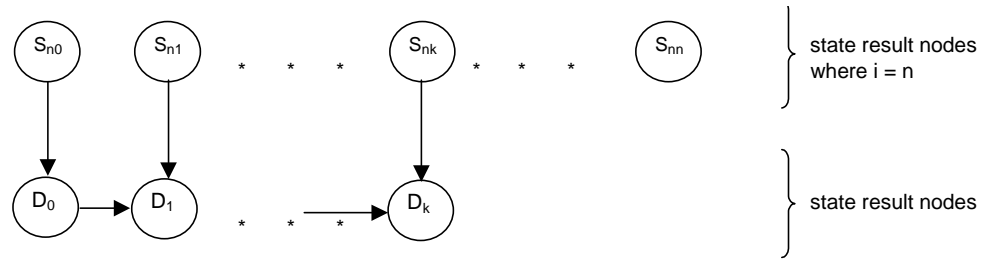


Figure 2.8 The State Result Nodes

**For Final node:** This node has 3 parent nodes. These are the last state result node, last edge node and last edge result node. It is only true if all parent nodes are true.

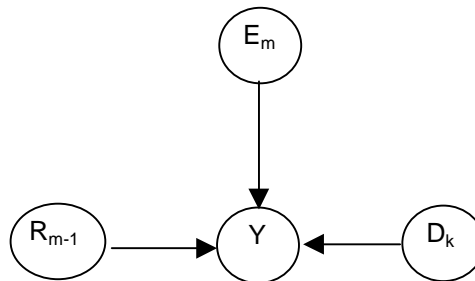


Figure 2.9 Final Node denoted by Y

There are no other nodes or edges for the BN.

### 2.2.2.3 The Probability Tables

**Probability table for vertex nodes:** There are 2 states for each node. And the states are probabilistic, they are not known. So, giving equal chances to each state seems

reasonable.

	$V_1$
TRUE	0.5
FALSE	0.5

Table 2.1 Probability table for vertex nodes

**Probability table for edge nodes:** There are exactly two parent nodes for each edge node. The probability table for edge  $E_i$  is as follows:

	$V_i$	TRUE		FALSE	
	$V_z$	TRUE	FALSE	TRUE	FALSE
$E_j$	TRUE	1	1	1	0
	FALSE	0	0	0	1

Table 2.2 Probability table for edge nodes

The table says that the node  $E_j$  with parents  $V_i$  and  $V_z$  is always true if any of the parent is true.

**Probability table for edge result nodes:** They have two parents nodes, except the first edge result node. The probability table is as follows:

	$E_j$	TRUE		FALSE	
	$R_{j-1}$	TRUE	FALSE	TRUE	FALSE
$R_j$	TRUE	1	0	0	0
	FALSE	0	1	1	1

Table 2.3 Probability table for edge result nodes

The table says that the node  $R_j$  with parents  $R_{j-1}$  and  $E_j$  is true if both of the parents are true. For node  $R_1$ , this table is not valid because there is no  $R_0$  exists. For  $R_1$   $Pr(R_1=True|E_1=True)=1$  and  $Pr(R_1=True|E_1=False)=0$

**Probability table for state nodes:** They have at most three parent nodes. If the parent node does not exist, it should be removed from the table. The table is as follows:



		TRUE				FALSE			
		TRUE		FALSE		TRUE		FALSE	
S <sub>ij</sub>	S <sub>i-1,j-1</sub>	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
		TRUE	0.5	1	0	0	0.5	0	1
	FALSE	0.5	0	1	1	0.5	1	0	1

Table 2.4 Probability table for state nodes

The state nodes are designed in order to count the number of true assignments to vertex nodes. If  $S_{ij}$  is true, then  $j$  of the first  $i$  state node is true. For the state nodes  $S_{nj}$ , the information gained is the total number of true vertex nodes.

**Probability table for state result nodes:** They have 2 parent nodes, except the first one. The probability table is as follows:

		TRUE		FALSE	
		TRUE	FALSE	TRUE	FALSE
D <sub>j</sub>	D <sub>j-1</sub>	1	1	1	0
		0	0	0	1

Table 2.5 Probability table for state result nodes

The table says that the result node  $D_j$  is true if any of the parents is true.

#### 2.2.2.4 An Example Transformation

To understand the model easily, let's define a vertex cover problem and solve it as an Exact Inference problem.

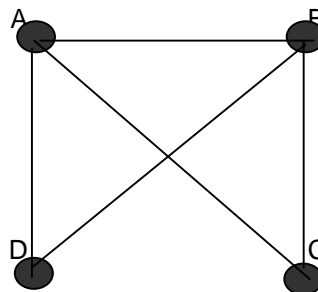


Figure 2.10 A VC Problem: Is it possible to have a vertex cover  $V$  where  $|V| \leq 2$ ?

Here, the problem is stated in Figure 2.10. We are supposed to give an answer to the

following question whether there is a vertex cover  $V$  where  $|V| \leq 2$ ? By using the transformation method proposed above the BN representation of this problem can be seen in Figure 2.11.

We can solve this problem by using a software. Then, what is the next step? How can we conclude about original VC problem?

If the probability for  $Pr(Y = TRUE) > 0$  then the answer to the vertex cover problem is yes, means it is possible to have a vertex cover with cardinality 2.

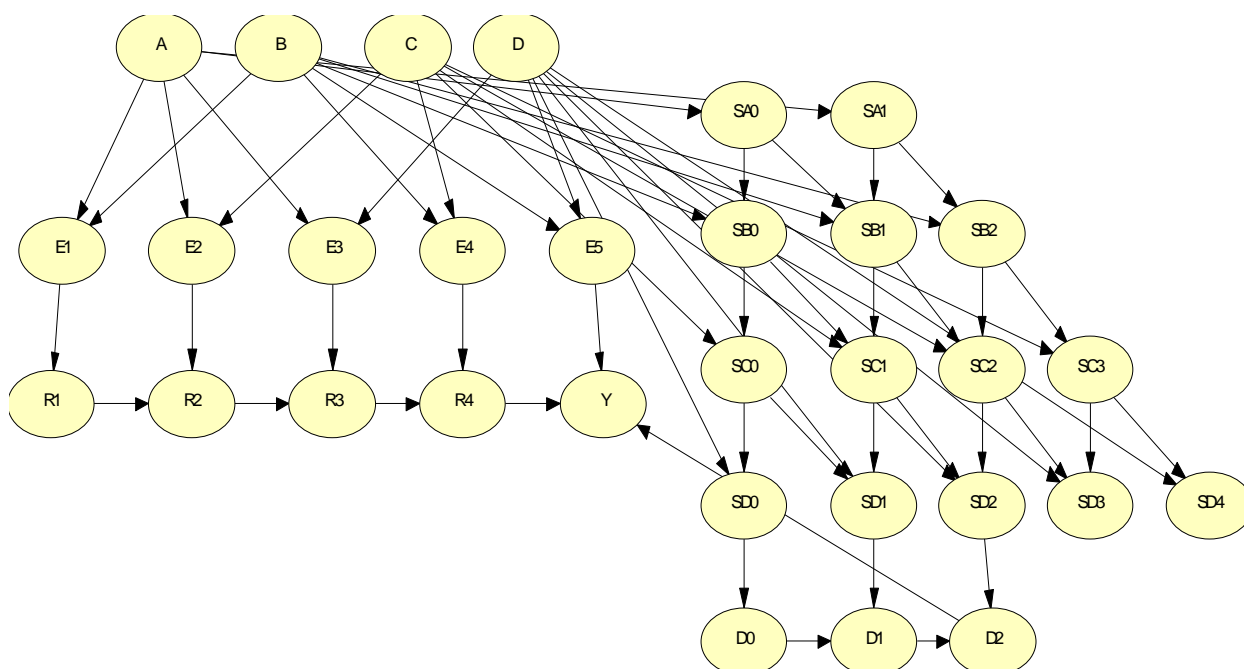


Figure 2.11 The BN representation of the example problem

After solving this simple problem with Hugin, the probability for  $Pr(Y = TRUE) = 0.0625$ . Then, the conclusion for the original VC problem is; we can find a vertex cover with  $|V| \leq 2$ .

### 2.2.2.5 Correctness Of The Transformation

**Lemma 1:**  $Pr(Y = TRUE) > 0$  if and only if there is a vertex cover with cardinality  $k$ .

**Proof:** If the states of all vertex nodes are given, the probability of  $Y$  is either 0 or 1. That is the result of having deterministic relations among the nodes. It is 1 if all incoming nodes for node  $Y$  is true. Then,  $R_{m-1}$ ,  $E_m$  and  $D_k$  is true.  $R_{m-1}$  and  $E_m$  is true iff all edges are

$$Pr(Y = TRUE) = \sum_{i=1}^{2^n} [Pr(Y = TRUE|evidence i) \cdot 1/2^n]$$

true. It is safe to say that all edges are true if  $Y$  is 1.  $D_k$  is true if the number of vertex nodes which is in state true, is smaller than or equal to  $k$  which is the cardinality number for the original VC. So, having  $Y=1$  means all edges are covered and the number of vertex nodes in state 1(true) is less than  $k$ . There are  $2^n$  possible assignments for vertex nodes. It means there are  $2^n$  different evidence options for vertex nodes.

where the evidence  $i$  is an assignment of vertex nodes different than evidence  $j$  where  $i \neq j$ . If the probability of  $Y$  is greater than 0, then for an evidence  $i$

$$Pr(Y = TRUE | evidence i)$$

is equal to 1. So, the original VC problem has a solution with cardinality  $k$ .

**Lemma 2:** For any BN representation with a given vertex cover, exactly  $n$  of the state nodes is true. For each group of state nodes, only one node is true.

**Proof:** By using induction

1- For the first group of states nodes,  $S_{10}$  and  $S_{11}$ , the probabilities are as follows:

$$Pr(S_{10}=True/V_1=True)=0 \text{ and } Pr(S_{10}=True/V_1=False)=1$$

$$Pr(S_{11}=True/V_1=True)=1 \text{ and } Pr(S_{11}=True/V_1=False)=0$$

So either  $S_{10}$  or  $S_{11}$  is true according to the condition of  $V_1$ .

It means for the first group of state nodes only one of them is true.

2- Think about the  $k^{th}$  group of state nodes  $S_{kj}$ .

Let  $S_{kz}$  is true, and all other state nodes in the group are false.

For the state nodes  $S_{k+1,j}$  the parents are  $S_{k,j-1}$ ,  $S_{k,j}$  and  $V_{k+1}$  if the parents exist. If  $S_{k,j-1}$  and  $S_{k,j}$  are false regardless of the value of  $V_{k+1}$   $S_{k+1,j}$  is false. So, in order to make  $S_{k+1,j}$  true either  $S_{k,j-1}$  or  $S_{k,j}$  must be true. For the nodes in the  $k^{th}$  group only  $S_{kz}$  is true, then the possible candidate nodes in the  $k^{th}$  group to be true are  $S_{k+1,z}$  or  $S_{k+1,z+1}$ . According to Table 2.4 only one of them is true conditioning to the value of  $V_{k+1}$ . Then we can conclude

that only one node in a group of state nodes is true.

If state node  $S_{ij}$  is true,  $j$  of the first  $i$  node is true. Let  $S_{ij}$  and  $S_{iw}$  is true. That means  $j$  of the first  $i$  node is true and  $w$  of the first  $i$  node is true. That is impossible if  $j \neq w$ . For each group of state nodes, only one node is true. There should be one true state node because in any case,  $j$  of  $i$  node(s) is true. Remember that  $j$  is from 0 to  $i$ . For each vertex node there is exactly one state group. Then, there are  $n$  groups of state nodes. If there should be one true node for each group, there should be  $n$  true state nodes in the whole BN.

**Lemma 3:** For the probability table of a state node, two parent state node can not be true simultaneously. So the probability assigned for both parents are true is not important.

**Proof:** That is related with Lemma 2. For each group of state nodes there is only one true state node. For a state node  $S_{ij}$  the parents  $S_{i-1,j}, S_{i-1,j-1}$  are in the same group. So, at most one of them is true.

**Lemma 4:** The transformation can be carried out in polynomial time.

**Proof:** The number of nodes for BN is  $(2m + k + \lceil (n+2) * (n + 1) / 2 \rceil)$ . For a complete graph  $m$  is  $n*(n-1)/2$ . So, the number of nodes seems to be reasonable. For each node the number of edges connecting the node with its parents are 0, 1, 2 or 3. So determining the edges is not a big deal. For the probability tables, the number of states of nodes are 2, the generic forms exist. So, it also doesn't increase the complexity. Finally, the transformation is  $O(n^2)$  and it is polynomial.

### 2.3 Solving the Exact Inference Obtained

In the literature there are some algorithms to solve Exact Inference problem. So we can use these algorithms to solve the Exact Inference problem for the special type of BN

obtained after the transformation. Consequently we will be able to read the solution of the Vertex Cover problem from the solution of the Exact Inference Problem. In order to solve this Exact Inference problem, the method must be chosen. We are interested in finding the probability of a single node. So, a method directly constructed to solve this node such as node elimination can be useful. If we want to reach to the result by giving initial values to vertex nodes, Junction tree algorithm can be useful. There are some interesting observations while using this algorithms.

### 2.3.1 Node Elimination Method

**Description:** Node Elimination Method is an algorithm to solve Exact Inference problem. It uses two fundamental operations. First is the barren node elimination and the second is arc reversal (Shacter, Ross, 1988). Barren node is the node that has no children and, not in the evidence set or the set of nodes whose posterior probability is asked. Arc reversal is the key step of the algorithm that changes the direction of the arcs in order to make the nodes barren and eliminate them. While converting the directions probability tables should be recalculated.

For the node elimination method, we initially remove the state nodes  $S_{ij}$  where  $j > k$ . These nodes don't have children and called as barren nodes. So, we can eliminate them.

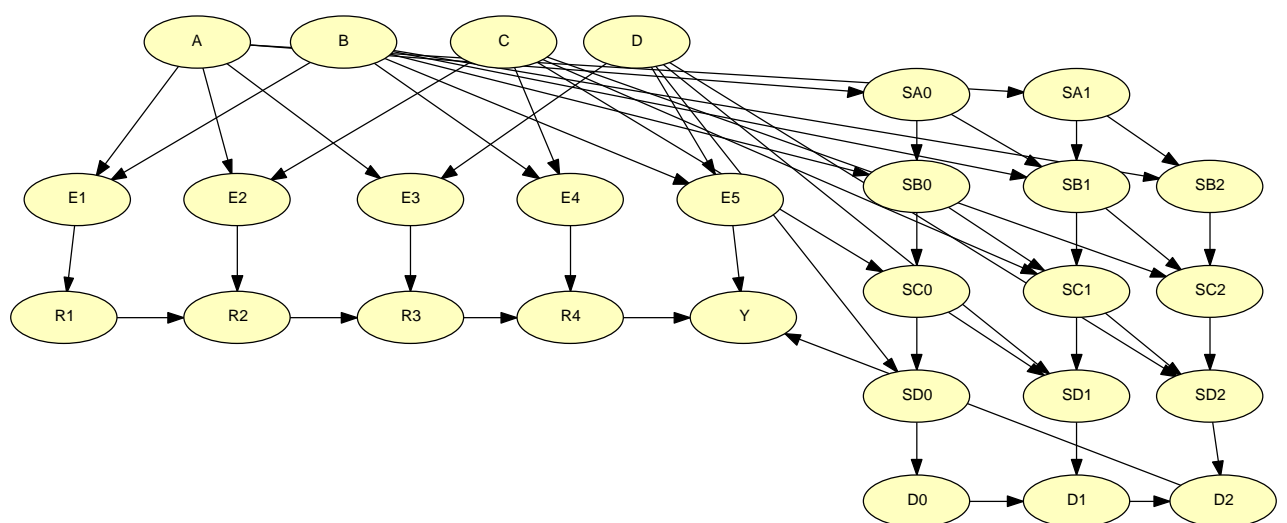


Figure 2.12 The BN after Elimination of Barren Nodes

By following our example, in Figure 2.12, the nodes SC3, SD3, SD4 are removed. After solving it, nothing changed for node  $Y$ . It is again 6.25 % for being true. The second step for this algorithm is arc reversal. Starting with the arc pointing to  $Y$  from  $D_k$ , we can reverse the arcs.

**Lemma 1:** By using a logical methodology, in each step of arc reversal a node will be barren and will be eliminated for the special type of BN obtained by the transformation.

**The methodology:** Starting with the arc reversal with the arc pointing from  $Y$  to  $D_k$ ,  $D_k$  will be barren. After this arc reversal  $D_{k-1}$  and  $S_{nk}$  will have a child which is  $Y$ . Applying arc reversal to the nodes which have only single child which is  $Y$  will yield a result that in each iteration a node will be barren, and in each step after elimination of the barren node, one or two nodes will be added to the set of the nodes which have a single child  $Y$ .

There are exactly  $n+2m+k+1+(n^2+3n)/2$  nodes. In the final graph we will have  $n+1$  nodes. Then, there are  $2m+k+(n^2+3n)/2$  arc reversals. While calculating the new probabilities, there is no need to calculate the node which will be a barren node. So at each iteration, we will only recalculate the probability table of node  $Y$ . The first step of arc reversal for the example problem is, reversing  $D2 \rightarrow Y$  to  $D2 \leftarrow Y$ .

After this conversion the graph will be as in Figure 2.13.

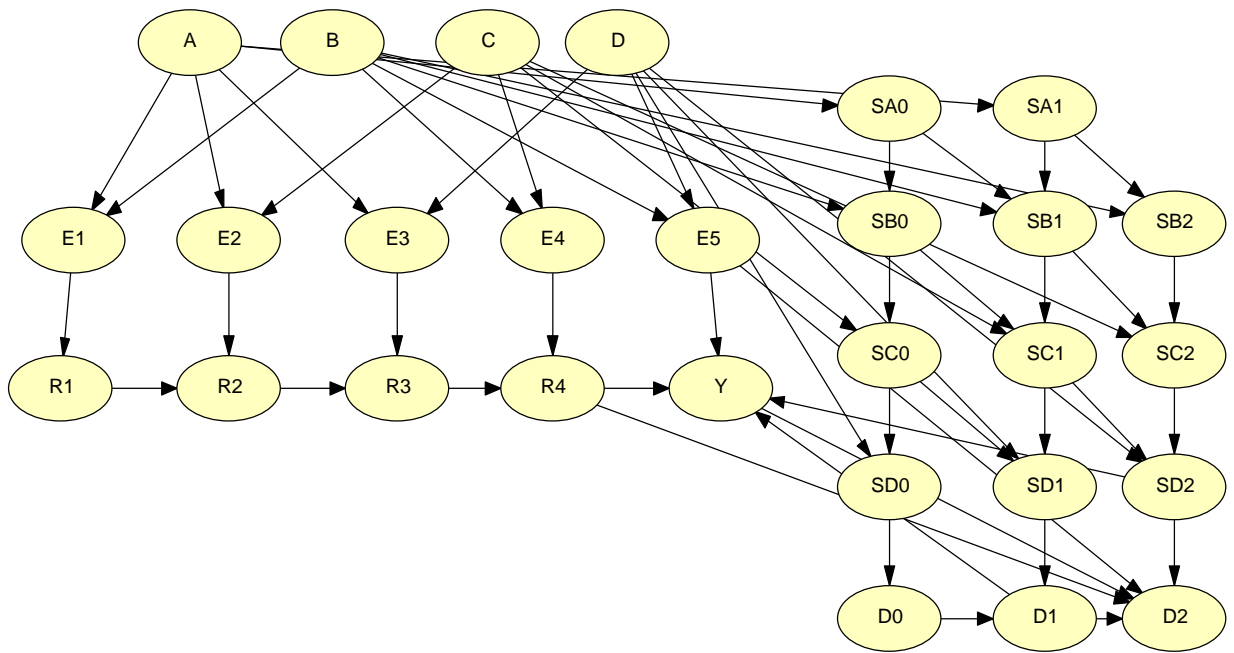


Figure 2.13 BN after conversion

Now, D2 is a barren node and there is no need to calculate the probability table for D2.

Calculating the probability table for Y is enough. The new table for Y is:

R4	T								F							
E5	T				F				T				F			
SD2	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F
D1	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F
Y	TRUE	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
	FALSE	1	0	0	1	1	1	1	1	1	1	1	1	1	1	1

Table 2.6 New Probability Table After Node Elimination

Finally after  $2m+k+(n^2+3n)/2$  arc reversals, the graph will be as in Figure 2.14.

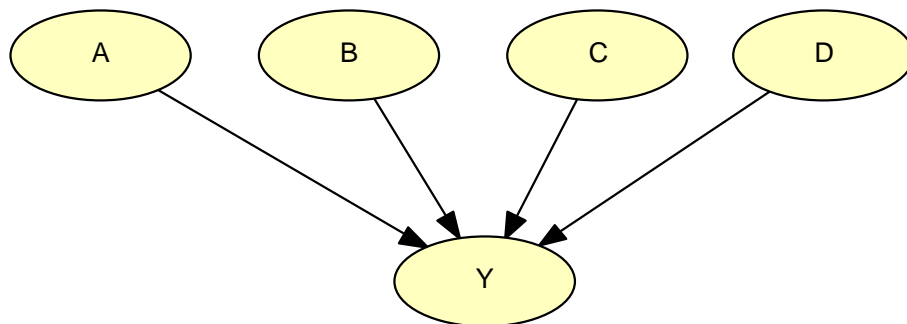


Figure 2.14 Final BN for Node Elimination Method

This network means, the state of  $Y$  depends on the states of  $A$ ,  $B$ ,  $C$  and  $D$ . By using the probability table of  $Y$  in this graph, we can conclude about any assignment for the vertex nodes. The table is as follows:

		T								F							
		T				F				T				F			
D		T		F		T		F		T		F		T		F	
		T	F	T	F	T	F	T	F	T	F	T	F	T	F		
Y	TRUE	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	FALSE	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1

Table 2.7 Final Probability Table after Node Elimination Method

This table gives us all possible assignments for the vertex.

### 2.3.2 Junction Tree Algorithm

**Description:** Junction Tree Algorithm is a method for performing probabilistic inference on a belief network. It works in two steps. First, a belief network is converted into a secondary structure. Then, probabilities of interest are computed by operating on that second structure. For repetitive queries this method has an advantage, and most of softwares are using this method.

This method is developed by Lauritzen and Spiegelhalter and refined by Jensen. Huang and Darwiche prepared a document for Junction Tree algorithm in order to implement the algorithm without additional help. All of the details of this algorithm and more can be found in that document (Darwiche, Huang,1994). As mentioned above, the algorithm consists of two stages. The first stage is graphical transformation as stated in Figure 2.15.



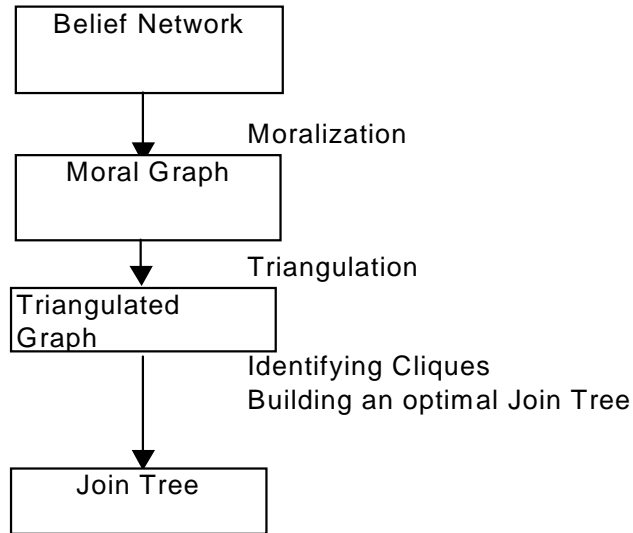


Figure 2.15 Graphical Modification of Junction Tree Algorithm

The BN we have for the vertex cover is not an ordinary BN. It has deterministic relations and the structure is fixed, so we can decompose the BN. While performing the Join Tree algorithm some simplifications can be performed in the first stage of graphical transformation which is the moralization step. In this step, parents of each node is connected with an edge. There are exactly  $n^2+2m+k$  marriages in the moralization step for our special network.

$m$  of the marriages are the marriages between vertex nodes. These marriages are done if the vertex are connected via an edge in the original VC problem graph. By marrying  $D_i$ 's and  $S_{n,i+1}$  we have  $k$  more arcs. By marrying  $E_i$ 's and  $R_{i-1}$ 's we introduce  $m-1$  arcs. 2 arcs come from  $E_m$  to  $D_k$ , and  $R_{m-1}$  to  $D_k$ .  $(n*(n-1))/2$  arcs are from the combination of  $S_{ij}$ 's and  $S_{i,j+1}$ 's where  $i$  is 1 to  $n-1$ .  $(n*(n+1))/2-1$  of them are the arcs between  $V_i$  and  $S_{i-1,j}$  where  $j$  is from 0 to  $i-1$ . If we sum up the numbers of the arcs added, it will give the number written above.

### 3.BENEFITS OF THE TRANSFORMATION

#### 3.1 Approximation algorithms of VC

As stated in previous sections, if there is a polynomial time transformation among two problems, heuristics developed in one side can be applied in the other side. In our case VC and Exact Inference problems are not polynomially equivalent, but VC is a special type of Exact Inference Problem. For an Exact Inference problem in the structure of the BN proposed, the approximation algorithms of VC can be applied to Exact Inference Problem.

The vertex cover problem can be described as an integer programming in the following way.

$$\begin{array}{ll} \min & \sum_{v \in V} x_v \\ \text{subject to} & x_u + x_v \geq 1, \quad (u, v) \in E \\ & x_v \in \{0, 1\}, \quad v \in V \end{array}$$

The LP-relaxation for this integer program is:

$$\begin{array}{ll} \min & \sum_{v \in V} x_v \\ \text{subject to} & x_u + x_v \geq 1, \quad (u, v) \in E \\ & x_v \geq 0 \quad v \in V \end{array}$$

A half-integral solution to LP-relaxation is a feasible solution in which each variable

is 0,1 or ½.

Any extreme point solution for the set of inequalities for this LP-relaxation is half-integral. This result leads to a factor 2 approximation for vertex cover: find an extreme point solution, and pick all vertices that are set to ½ or 1 in this solution. (Vazirani,2001)

This approximation can be used for the BN we achieved after polynomial time transformation from VC problem to Exact Inference problem. If the deal for a BN introduced in previous chapters is whether the  $P(Y=true)>0$  or equal to 0, the answer can be given by using the LP-relaxation above.

After solving the LP-relaxation,  $V_i$  is true if  $V_i$  is inherited from  $X_i$  and  $X_i$  is 1 or ½. Set  $V_i$  false, if  $X_i$  is 0. Calculating the probability for node  $Y$  is very easy, and can be done in polynomial time, after assigning the values of the chances nodes in the Belief Network. Because, all other nodes are deterministic nodes.

$$a = \sum_{i=1}^n \text{ceil}(X_i) \quad \text{where}$$
$$\sum_{i=1}^n X_i = \text{min of LP-relaxation and}$$
$$X_i \in \{0, 1, 1/2\} \text{ and}$$
$$\text{ceil}(X_i) = \lceil X_i \rceil$$

Here,  $a$  is the variable for comparison with  $k$ . It is known that the lower bound solution for the optimal vertex cover problem is  $a/2$ .

If  $a < k$  then the probability of  $Y = \text{true}$  is greater than 0.

If  $a/2 < k < a$  then to come up with an idea about probability of  $Y$  at this time is impossible.

If  $k < a/2$  then the probability for  $Y = \text{true}$  is 0.

All of the operations above can be done in polynomial time. So, it is possible to come up with an idea about the probability of  $Y$  in polynomial time.

Having a better approximation, scheme and improving on factor 2 for VC is an open problem. There are other methods that give factor 2 approximation for VC problem.

0- Set  $C$  to empty set.

1- Let  $E'$  is the copy of the graph  $G$ .

2- Remove any edge  $(u,v)$  from the graph  $E'$  if exists. If not stop.

3- Remove all other edges adjacent to  $(u,v)$  from  $E'$ .

4- Add  $u$  and  $v$  to set  $C$ .

5- Goto step 2.

When this algorithm stops, the edges removed in step 2 have no common edge. So the size of the optimal solution set  $|C^*|$  should exceed  $|A|$  where  $A$  is the set of removed edges. The set  $C$  is a vertex cover, and  $|C| = 2|A|$ . So this approximation is also factor 2. Instead of dealing with linear programming, this easy and simple procedure may help to come up with an idea about the probability of  $Y$  by using the same idea above.

A lot of people try to find a simpler and improved algorithms for vertex cover problem. (Chen, Kanj, Jia, 1999) proposes a new algorithm that solves the problem in  $O(kn + 1.271^k k^2)$ . This is also true for the Exact Inference problem in the structure of the transformed VC problem.

Vertex Cover Problems are deeply worked, so special types of graphs for VC can be solved in polynomial time. If the graph is a tree, an algorithm that solves the problem with complexity  $O(n+m)$  exists with a dynamic programming approach.

There are also approximation algorithms for the inference problem. It is possible to use the approximate inference problem solving techniques for the Vertex Cover problem. In the next section results of the experiments for approximate inference as a heuristic for VC can be found.

### 3.2 Using softwares developed for Inference Problem as a VC Solver

The aim of this section is to find out whether it is possible to discover a better approximation algorithm for VC by using the approximation techniques for Exact Inference and the transformation proposed in Chapter 2 from a practical point of view.

The process consists of 3 steps.

- 1- Random VC problem inference creation
- 2- Converting VC to BN
- 3- Finding the optimal and approximate solution

Coding for first and second step is done by GSAMS, a visualized algorithm modelling system. The third needs using the Application Program Interface (API) of softwares developed by BN specialists. For performance reasons, Hugin and MSBN are used and the coding is done by C++.

#### 3.2.1 Softwares Used

For solving the problem, the following software tools are used.

**HUGIN:** The HUGIN System is a tool enabling you to construct model based expert systems in domains characterized by inherent uncertainty. The models supported are Belief Networks and its extension influence diagrams. The HUGIN System allows you to define both discrete nodes and to some extent continuous nodes in your models.

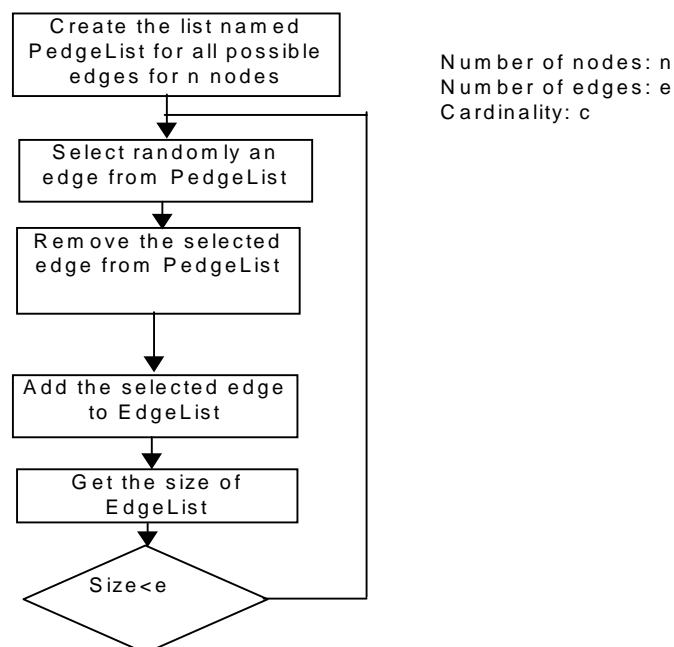
You have the opportunity to use the HUGIN System through HUGIN RunTime an easy-to-use graphical environment. You can also use the HUGIN API which comes as a library for C (or C++).

**MSBN:** MSBNx is a component-based Windows application for creating, assessing, and evaluating Belief Networks, created at Microsoft Research. The application's installation module includes complete help files and sample networks. Belief Networks are encoded in an XML file format. The application and its components run on Windows 98,

Windows 2000, and Windows XP. MSBNx inferential operations provide both inference about states of inference and about the value of information for unobserved evidence. The services and modeling environment supports both diagnostic and troubleshooting mingles observations and repair operations. MSBNx facilitates the development and use of new *add-in* components. The modeling environment provides a means for assessing distinctions and beliefs, and special interfaces and tools for representing the asymmetric nature of probability distributions.

### 3.2.2 Random VC Creation

Firstly, random VC problem is created by the following algorithm. The number of nodes, edges and cardinality of the problem are the input data for the algorithm. The algorithm is as follows:



### **3.2.3 Converting VC to BN**

The second step is converting the random VC problem to a BN. For Hugin the text representation of BN called NET language is used. For MSBN there are two different representation techniques, one is the XML representation and the other is text representation. The text representation is preferred in order to use the work done for Hugin and DSC language is used. The VC problem is converted to an Exact Inference problem by the transformation proposed in Chapter 2.2.2 .

### **3.2.4 Solving the Problem**

The third step is solving the problem. The optimal solution for the BN and approximate solution for different epsilon values are obtained by the algorithm used by the software Hugin 5.1. MSBN only finds the exact solution.

The number of nodes, edges are cardinality for the vertex cover problem is an input. The summary result of whole experiment can be seen in Table 3.1.

### **3.2.5 Results**

In Table 3.1. NOE denotes number of edges, NON denotes number of nodes and C denotes the cardinality of the problem.

“Exact” shows the exact solution for node  $Y$  in state “True”. The values starting with “eps” demonstrates the result of the approximation algorithm for node  $Y$  for the same node where epsilon value is the stated value. “cf” is compilation failure means no solution is available.

The compilation failure is about the algorithm used by the program. The error message is like “inconsistency\_or\_underflow” and the description is “Propagation of

inconsistent evidence has been attempted, or perhaps (but unlikely) underflow has occurred”.

50 different instances have been tried. Table 3.1 shows the whole results.

NOE	NON	C	Exact	Eps 0.1	Eps 0.2	Eps 0.3	Eps 0.4	Eps 0.5	Eps 0.6	
2	3	2	0.5			0.5	0.5	0.5	1	→ 1
2	3	2	0.5			0.6		0.75		→ 1
3	3	2	0.375			0.4	1	1	1	→ 1
3	3	2	0.375			0.4	1	1	1	→ 1
4	4	2	0.125	0.125	0.2222	0	0	0	0	cf
4	4	4	0.4375			0.5	1	1	1	cf
4	4	4	0.4375			0.5	1	1	1	cf
6	4	4	0.3125		0.333	0				cf
6	4	4	0.3125		0.1667	0				cf
6	5	4	0.2814			0.2857				
6	5	4	0.2814			0.2222				
10	5	2	0	0	0	0				
10	5	4	0.1563	0.0909	0	0	0	0	0	cf
8	6	2	0	0	0	0	0	0	0	cf
8	6	6	0.25			0.1538				
8	6	6	0.25	0.22	0.1563	0.1538	cf			
10	6	6	0.1875	0.1875	0	0				
9	7	5	0.1484	0.1379	0.1875	0				
9	7	5	0.1406		0.0857	0				
9	7	5	0.1406		0.0417	0	0	0		cf
15	8	2	0	0	0	0				
15	8	6	0.0742	0.0133	0	0	0	0	cf	
15	8	6	0.0664	0	0	0				
15	8	6	0.0781	0	0	0				
6	9	4	0.0391	0	0	0	0			cf
6	9	4	0.0078	0	0	0	0			cf
6	9	5	0.1095		0.1011	0				
6	9	9	0.2578			0.2278	0	0		cf
6	9	9	0.2813			0.1351	1			cf
24	12	6	0	0	0	0				
24	12	10	0.0026	0	0					
24	12	11	0.1546	0.0306	0					
24	12	12	0.228	0.116	0.0992	0				
25	12	12	0.2145	0.1149	0.0148	0.0196	0			
26	12	12	0.1393	0.1146	0.0077	0				
40	12	11	0.0242	0.0089	0					
40	12	12	0.0898	0.0521	0					
25	14	6	0	0	0					
25	14	7	0	0	0					
25	14	8	0.0033	0	0					
25	14	9	0.0882	0.0121	0					
25	14	10	0.1116	0.0538	0					
25	14	11	0.1445	0.0519	0					
25	14	12	0.2444	0.2221	0.115	0.0171	0			
25	14	13	0.2832	0.2075	0.1					cf
25	14	14	0.314	0.2542	0.1272	0.0026	0			
40	14	10	0.0883	0						
25	15	6	0	0	0					
40	15	9	0	0						
40	15	10	0.0813	0						

Table 3.1 Results of Experiment



As it is seen from the Table 3.1, number of nodes for the vertex cover does not exceed 15. Both of the software used for solving inference problem can not solve the VC with more than 15 nodes. Hugin stopped with a message denotes that no memory is available. MSBN stopped without giving any message after a work lasting 3 hours.

### 3.3 Analysis of The Result

These results show that if the optimal solution is very near to 0 but not 0, then the approximation algorithm fails. However, if the optimal solution is greater than 0.4 the epsilon value can be greater than 0.5(Experimental result). It is known that there exist polynomial time algorithms which calculates the Inference problem with epsilon greater than 0.5. Then, the conditions when the exact solution is greater 0.4 becomes an important question. Investigating the ratio of number of nodes  $n$  versus number of edges  $m$  for cardinality  $n$  can be useful to answer to this question. Table 3.2 shows this type of demonstration with the same result.

NCE	NON	C	NON/NOE	C/NON	Exact	Eps 0.1	Eps 0.2	Eps 0.3	Eps 0.4	Eps 0.5	Eps 0.6
40	12	12	0.3	1	0.0596	0.0521	0				
35	12	12	0.461538	1	0.1393	0.1146	0.0077	0			
25	12	12	0.48	1	0.2145	0.1149	0.0148	0.0196	0		
24	12	12	0.5	1	0.228	0.116	0.0992	0			
25	14	14	0.56	1	0.314	0.2542	0.1272	0.0026	0		
10	6	6	0.6	1	0.1875	0.1875	0	0			
6	4	4	0.666667	1	0.3125		0.333		0	cf	
6	4	4	0.666667	1	0.3125		0.1667		0	cf	
8	6	6	0.75	1	0.25			0.1538		cf	
8	6	6	0.75	1	0.25	0.22	0.1563	0.1538		cf	
4	4	4	1	1	0.4375			0.5		1	1
4	4	4	1	1	0.4375			0.5		1	1
6	9	9	1.5	1	0.2578			0.2278		0	0
6	9	9	1.5	1	0.2813			0.1351		1	cf

Table 3.2 Results for C/NON = 1

As seen from the figure the exact value increases for NON=12 as NOE decreases. Then, the ratio has a meaning and if it is very high, it is possible to find a polynomial time algorithm which gives the approximate solution. Another observation that can be obtained from the table is the directly related with the number of nodes. If it increases, the number of different instances of the vertex nodes of BN increases. So, it decreases the value of exact

solution. We can easily conclude that for small networks giving more accurate answers is easier.

In our sample, a polynomial time approximate inference algorithm can give true answer for the vertex cover problem with 23 %. It gives the wrong result with 3 %. It does not give an answer because of compilation error with 74 %. If the results are achieved after the implementation of the approximation algorithm with epsilon 0.6, it will be safe to say that the true result for the vertex cover problem is obtained with probability 0.88.

These results are not sufficient to say that a better approximation algorithm for Vertex Cover is found. Because of technological constraints, the number of nodes in the experiment data is very restricted and the results are not good.

## **4. CONCLUSION & FUTURE WORK**

The main contribution of this study is providing an alternative polynomial transformation that shows that the Exact Inference problem is NP-complete. The transformation uses the well known vertex cover problem. This polynomial transformation enabled us to try to solve vertex cover problem by using algorithms proposed in the literature for the Exact Inference Problem. The results of these experiments were not very encouraging.

Actually, proving that the Exact Inference problem is NP-Complete is not a new result, because it was proven in a different way before. However, proving the relation between VC and Exact Inference can be useful since Vertex Cover Problem has various applications and very well studied. The fact that the Vertex Cover problem is a special case of Exact Inference problem can be useful in different ways. For instance, any development, a better algorithm from a practical point of view for general type Exact Inference problem can be beneficial for the solving VC problem.

As a future work, it is possible to convert easy problems to BN structure, and find the instances of BNs where the solution of Exact Inference is polynomial. Another method to find instances of BNs where the solution of Exact Inference is polynomial can be converting the special VC problems, for which there are polynomial time algorithms in the literature to Exact Inference Problem.

In section 2.3, to solve the Exact Inference problem in a specific structure, some methods are suggested. However, as a future work more useful methods can be found. Especially, for Junction Tree Algorithms, finding cliques is very important, and our special network has a potential for finding a better method to determine the cliques from the existing methods.

As the results of the experiments show, although the transformation is polynomial, number of nodes for BN is very high, and solving the Exact Inference is very difficult. For both type of problems, VC and Exact Inference, there is no general efficient approximation algorithm. Our approximation algorithm suggestion for VC by leaning on the transformation has also failed.

There exists softwares for solving Vertex Cover. However, these are not well known and commercial products. Using MSBN or Hugin as a VC solver after eliminating the technological constraints is another side of the transformation. In the current technological condition, Hugin can only solve VC problem with 15 nodes, but as the capacity of computers increase in the future, the number of nodes solved by Hugin or MSBN will certainly increase.

## 5.REFERENCES

1. Abdelbar Ashraf M. , “The Complexity of approximating MAPs for belief networks with bounded probabilities”, *Artificial Intelligence* 124, pages 283-288, 2000.
2. Andrasfai B., *Graph Theory: Flows, matrices*, Adam Hilger, New York, 1991.
3. Batchelor, C. , “Application of Belief Networks to water management studies”, *Agricultural Water Management* 40, pages 51-57, 1999.
4. Bilgic T. , *Lecture Notes of IE 522 Decision Analysis*, 2002.
5. Chen J., Kanj I. A., and Jia W., “Vertex cover: further observations and further improvements.” *Proceedings of the 25th International Workshop on Graph-Theoretical Concepts in Computer Science (WG'99) Lecture Notes in Computer Science* 1665, pages 313-324, Springer-Verlag, New York, 1999
6. Cooper Gregory F. , “The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks”, *Artificial Intelligence* 42, pages 393-405, 1990.

7. Cowell Robert G. , *Probabilistic Networks and Expert Systems*, Springer, New York, 1999.
8. Dagum P. and Luby M. , “Approximating probabilistic inference in Bayesian belief networks is NP-Hard”, *Artificial Intelligence* 60, pages 141-153, 1993.
9. Dagum P. and Luby M. , “An optimal approximation algorithm for Bayesian inference”, *Artificial Intelligence* 93, pages 1-27, 1997.
10. Garey Michael R. and Johnson David S. , *A Guide to the Theory of NP-Completeness*, W.H Freeman and Company, New York, 1979.
11. Geiger D. and Heckerman D. , “Knowledge representation and inference in similarity networks and Bayesian multinetts”, *Artificial Intelligence* 82, pages 45-74, 1996.
12. Goodaire Edgar G. and Permenter Michael M., *Discrete Mathematics with Graph Theory*, Prentice Hall, Upper Saddle River, 1998.
13. Guoxing Y. , “Transformation of multidepot multisalesman problem to standart travelling salesman problem”, *European Journal of Operations Research* 81/3, pages 557-560, 1995.
14. Homer S. and Selman A., *Computability and Complexity Theory*, Springer, New York, 2001.
15. Howard R. , "From Influence to Relevance to Knowledge", in R. M. Oliver and J. Q. Smith (eds), *Influence Diagrams, Belief Nets and Decision Analysis*, Chapter 1, pages. 2-23, 1990.
16. <http://www.cs.berkeley.edu/~murphyk/Bayes/bayes.html>

17. <http://research.microsoft.com/adapt/MSBNx/MSBNxManual.asp>
18. Huang C. and Darwiche A. , “Inference in Belief Networks: A Procedural Guide”, *International Journal of Approximate Reasoning* 11, pages 1-158, 1994.
19. HUGIN API Reference Manual Version 3.1, 1997.
20. Jungnickel D. , *Graphs, Networks and Algorithms*, Springer, New York, 1999.
21. Papadimitriou Christos H. , *Combinatorial Optimization, Algorithms and Complexity*, Prentice-Hall, New York, 1982.
22. Reeves Colin R. , *Modern Heuristic Techniques for Combinatorial Problems*, Mc Graw-Hill, London, 1995.
23. Roth D., “On the Hardness of Approximate Reasoning”, *Artificial Intelligence* 82, pages 273-302, 1996.
24. Shachter and Ross D. , “Probabilistic Inference and Influence Diagrams”, *Operations Research* 36/4, pages 586-602, 1988.
25. Shimony S.E. , “Finding MAPs for belief networks is NP-Hard”, *Artificial Intelligence* 68, pages 399-410, 1994.
26. Vazirani V. , *Approximation Algorithms*, Springer, Berlin, 2001.

## 5.REFERENCES

1. Abdelbar Ashraf M. , “The Complexity of approximating MAPs for belief networks with bounded probabilities”, *Artificial Intelligence* 124, pages 283-288, 2000.
2. Andrasfai B., *Graph Theory: Flows, matrices*, Adam Hilger, New York, 1991.
3. Batchelor, C. , “Application of Belief Networks to water management studies”, *Agricultural Water Management* 40, pages 51-57, 1999.
4. Bilgic T. , *Lecture Notes of IE 522 Decision Analysis*, 2002.
5. Chen J., Kanj I. A., and Jia W., “Vertex cover: further observations and further improvements.” *Proceedings of the 25th International Workshop on Graph-Theoretical Concepts in Computer Science (WG'99) Lecture Notes in Computer Science* 1665, pages 313-324, Springer-Verlag, New York, 1999
6. Cooper Gregory F. , “The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks”, *Artificial Intelligence* 42, pages 393-405, 1990.



7. Cowell Robert G. , *Probabilistic Networks and Expert Systems*, Springer, New York, 1999.
8. Dagum P. and Luby M. , “Approximating probabilistic inference in Bayesian belief networks is NP-Hard”, *Artificial Intelligence* 60, pages 141-153, 1993.
9. Dagum P. and Luby M. , “An optimal approximation algorithm for Bayesian inference”, *Artificial Intelligence* 93, pages 1-27, 1997.
10. Garey Michael R. and Johnson David S. , *A Guide to the Theory of NP-Completeness*, W.H Freeman and Company, New York, 1979.
11. Geiger D. and Heckerman D. , “Knowledge representation and inference in similarity networks and Bayesian multinetts”, *Artificial Intelligence* 82, pages 45-74, 1996.
12. Goodaire Edgar G. and Permenter Michael M., *Discrete Mathematics with Graph Theory*, Prentice Hall, Upper Saddle River, 1998.
13. Guoxing Y. , “Transformation of multidepot multisalesman problem to standart travelling salesman problem”, *European Journal of Operations Research* 81/3, pages 557-560, 1995.
14. Homer S. and Selman A., *Computability and Complexity Theory*, Springer, New York, 2001.
15. Howard R. , "From Influence to Relevance to Knowledge", in R. M. Oliver and J. Q. Smith (eds), *Influence Diagrams, Belief Nets and Decision Analysis*, Chapter 1, pages. 2-23, 1990.
16. <http://www.cs.berkeley.edu/~murphyk/Bayes/bayes.html>

17. <http://research.microsoft.com/adapt/MSBNx/MSBNxManual.asp>
18. Huang C. and Darwiche A. , “Inference in Belief Networks: A Procedural Guide”, *International Journal of Approximate Reasoning* 11, pages 1-158, 1994.
19. HUGIN API Reference Manual Version 3.1, 1997.
20. Jungnickel D. , *Graphs, Networks and Algorithms*, Springer, New York, 1999.
21. Papadimitriou Christos H. , *Combinatorial Optimization, Algorithms and Complexity*, Prentice-Hall, New York, 1982.
22. Reeves Colin R. , *Modern Heuristic Techniques for Combinatorial Problems*, Mc Graw-Hill, London, 1995.
23. Roth D., “On the Hardness of Approximate Reasoning”, *Artificial Intelligence* 82, pages 273-302, 1996.
24. Shachter and Ross D. , “Probabilistic Inference and Influence Diagrams”, *Operations Research* 36/4, pages 586-602, 1988.
25. Shimony S.E. , “Finding MAPs for belief networks is NP-Hard”, *Artificial Intelligence* 68, pages 399-410, 1994.
26. Vazirani V. , *Approximation Algorithms*, Springer, Berlin, 2001.